Titre: Title:	Stimulations combinées dédiées au rétablissement de l'évacuation chez les patients souffrant de dysfonctions urinaires
Auteur: Author:	Aguibou Ba
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ba, A. (2004). Stimulations combinées dédiées au rétablissement de l'évacuation chez les patients souffrant de dysfonctions urinaires [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7271/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7271/
Directeurs de recherche: Advisors:	
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

STIMULATIONS COMBINÉES DÉDIÉES AU RÉTABLISSEMENT DE L'ÉVACUATION CHEZ LES PATIENTS SOUFFRANT DE DYSFONCTIONS URINAIRES

AGUIBOU BA

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

JANVIER 2004

Your file Votre référence
ISBN: 0-612-91932-3
Our file Notre référence
ISBN: 0-612-91932-3

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

STIMULATIONS COMBINÉES DÉDIÉES AU RÉTABLISSEMENT DE L'ÉVACUATION CHEZ LES PATIENTS SOUFFRANT DE DYSFONCTIONS URINAIRES

Présenté par : BA Aguibou
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. BRAULT Jean-Jules, Ph.D ,président
M. SAWAN Mohamad, Ph.D., membre et directeur de recherche
Mme CHERIET Farida, Ph.D, membre

DÉDICACE

À mes parents, Aminata Sall et Ibrahima Ba,
Mon frère et ma sœur, Karimou et Hadja
Je vous aime.

REMERCIEMENTS

Je tiens avant tout à remercier mon directeur de recherche, Mohamad Sawan, qui a bien voulu m'accepter au sein de son équipe de recherche, pour ses conseils, ses suggestions ainsi que son support tout au long de ma maîtrise.

Un remerciement particulier à Grace pour sa présence ainsi que son soutien moral lors de ces deux dernières années.

Tous mes remerciements,

À Gaétan Décarie, technicien du DGE de l'ÉPM, pour son grand support technique ainsi que sa constante disponibilité, à Jonathan Coulombe et à Tommy Desilets pour leur support, respectivement lors de la réalisation de la puce, et lors de la validation de l'implant et la rédaction de ce mémoire.

À Madame Farida Cheriet et Monsieur Jean-Jules Brault, professeurs à l'École Polytechnique de Montréal, pour bien avoir voulu participer au jury d'examen de ce mémoire.

À mes amis ici ou à l'extérieur du continent ainsi que ma famille à Dakar et Nouakchott.

À tous les techniciens et techniciennes du département de génie électrique de l'École Polytechnique et mes collègues de l'équipe PolySTIM.

Enfin, je remercie la Société Canadienne de Microélectronique pour les outils et le support lors de la conception et la fabrication du circuit intégré.

RÉSUMÉ

Les travaux présentés dans ce mémoire trouvent leur application dans le domaine de la récupération des fonctions du système urinaire. Nous présentons des dispositifs implantables utilisés pour le rétablissement des fonctions de la vessie chez les patients paraplégiques, ceux atteints de maladies congénitales ou ceux ayant perdu le contrôle volontaire de leur vessie suite à une lésion de la moelle épinière.

Nous présentons tout d'abord une nouvelle version d'un implant dédié réalisé sur un circuit imprimé de moins de 3 cm de diamètre avec des composants discrets disponibles commercialement. Cet implant est la version améliorée d'un système déjà fonctionnel. Placé sous la peau d'un patient, il est en mesure de réaliser deux types de stimulations pour la récupération des fonctions urinaires. Une stimulation sélective est appliquée pour rétablir la miction volontaire tandis qu'une fonction de stimulation permanente permet de réduire les problèmes d'hyperréflexie (contractions indésirées de la vessie). Le mode permanent est une stimulation à très basse fréquence et très basse amplitude alors que la stimulation sélective est ponctuelle, précise, à plus forte amplitude. Un contrôleur externe envoie à l'implant de façon transcutanée l'énergie ainsi que les paramètres de stimulation à l'aide d'un lien inductif opérant à haute fréquence. Ce stimulateur fonctionne avec deux tensions d'alimentation distinctes; l'une à 5V pour la stimulation sélective qui nécessite une tension élevée, et une pile de 3.3V dédiée à la stimulation permanente.

Des études expérimentales in vivo, afin de caractériser la fiabilité et la fonctionnalité du stimulateur, ont été menées sur des chiens à l'animalerie de l'Université McGill. Les résultats obtenus à partir des tests en laboratoire et des expérimentations chez l'animal avec les versions précédentes du système avaient déjà démontré le fonctionnement des méthodes de stimulation sélective et de stimulation permanente. Elles avaient aussi montré la pertinence de ces techniques, applicables sur le plan médical aussi bien au domaine de la réhabilitation vésicale qu'à d'autres types de réadaptation. Au moment de la rédaction de ce mémoire, d'autres expérimentations sont en cours afin de parfaire les techniques de stimulation et de valider les améliorations apportées au système.

Un circuit intégré dédié a été réalisé et testé dans la technologie CMOS 0.18um. En plus de posséder toutes les fonctionnalités des stimulateurs précédemment réalisés en technologie discrète, il combine un nouveau type de stimulation sélective dite flexible permettant à l'usager de générer les formes d'ondes de stimuli de son choix. Le circuit contient aussi un module de mesure d'impédance permettant de caractériser l'interface électrode-nerf. Le design combine des fonctions numérique et analogique et occupe 1300x1300um^2 sans les plots de la puce. Le système a été conçu de façon à être aisément reprogrammable et ses deux canaux de sortie lui confère une flexibilité accrue.

Les tests ont démontré que les hautes fréquences générées pouvaient atteindre 75 KHz tandis que les basses variaient de 4.6 Hz à 1.2 KHz. Le système de stimulation peut générer des durées d'impulsions avec une précision de 3 us. 32 valeurs d'amplitude de 8

bits chacune peuvent être placés dans la RAM, offrant ainsi une grande variété de formes d'ondes de stimulation.

Une erreur est survenue lors de l'interconnexion des blocs analogique et numérique, causant ainsi une impossibilité de varier l'amplitude des stimuli de sortie, cependant la fonctionnalité globale du système a été vérifiée. De plus, une version améliorée de cette puce est en cours de fabrication par la Société Canadienne de Microélectronique.

ABSTRACT

The work presented in this master thesis is dedicated to the rehabilitation of the urinary system. The reported implantable devices allow to recuperate the bladder functions of paraplegic patients, those having lost the voluntary control of their bladder following a lesion of the spinal cord or those reached of diseases.

First, we present a new version of a dedicated implant mounted on a less than 3 cm diameter printed circuit board (PCB) with discrete components all commercially available. This implant, placed under the patient's skin, produces two types of stimulation techniques for the recovery of the urinary functions. A selective stimulation is applied to restore the voluntary micturition while a permanent stimulation technique makes possible the reduction of the hyperreflexia symptoms (unsolicited contractions of the bladder). The permanent mode is a very low frequency and very low amplitude stimulation, while the selective stimulation is more specific, and consists of a series of bi-frequency stimuli with higher amplitude. An external controller sends transcutaneously the needed power as well as the stimulation parameters through a high frequency inductive link. The stimulation system operates with two distinctive voltages; one at 5 V for the selective stimulation which requires high voltage, and an embedded battery of 3.3V is dedicated to permanent stimulation.

In vivo experimental studies, in order to characterize the reliability and the functionalities of the stimulator, were undertaken on dogs at the McGill University. The

results obtained from the circuits test and the experiments on dogs, with the previous versions of the system, had already shown the effectiveness of both selective and permanent stimulation methods. They had also shown the importance of these techniques, on a medical point of view, in fields not limited only to the bladder rehabilitation but also to others types of rehabilitation techniques. Also, others experiments are being conducted in order to improve the stimulation techniques and to validate the new version of the implant.

On the other hand, a dedicated integrated circuit was designed, fabricated and tested in CMOS 0.18um technology. In addition to having all the functionalities of our previous stimulation system realized in discrete technology, the new stimulator combines a novel type of selective stimulation called flexible stimulation. This mode allows the generation of a multitude of waveforms stimuli. The circuit contains also an impedance measurement building block making possible the characterization of the electrode-nerve interface. The design combines digital and analog blocks and occupies 1300x1300um^2 excluding the pads. The system was designed in order to be easily reprogrammable and its two output channels confers it an increased flexibility.

The tests showed that the generated high frequencies could reach 75 KHz while the low ones varied from 4.6 Hz to 1.2 KHz. The stimulation system can generate pulse widths with a precision of 3 us. 32 8-bits magnitude values can be placed in a register, thus offering a large variety of stimulation waveforms.

An error occurred while connecting the analog and the digital blocks, causing an impossibility to vary the amplitude of the delivered stimuli. However, the global functionalities of the system were demonstrated. Furthermore, a new integrated circuit is under fabrication to improve the design and correct the described dysfunction.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ANNEXES

LISTE DES SIGLES ET ABRÉVIATIONS

AM	Amplitude modulation
Amp	Amplitude
ASIC	Application specific integrated circuit
BICMOS	Bipolar complementary metal-oxide semiconductor
BCS	Bipolar current source
CAN	Convertisseur analogique à numérique
CMC	Canadian microelectronics corporation
CMG	Cystometrogram
CMOS	Complementary metal-oxide semiconductor
CNA	Convertisseur numérique à analogique
CPF	Contrôleur portatif flexible
CPS	Contrôleur portatif simplifié
CRC	Cyclic redundancy check
CTF	Contrôleur de test flexible
DAC	Digital to analog converter
DH	Detrusor hyperreflexia
EMG	Electromyogram
EPROM	Erasable programmable read only memory
FPGA	Field programmable gate array
Freq	Frequency
FSM	Finite state machine
HFA	High frequency amplitude
HFP	High frequency period
HFW	High frequency width

IVU	Intravenous urography
LCD	Liquid crystal display
LFA	Low frequency amplitude
LFP	Low frequency period
LFW	Low frequency width
LSK	Load shift keying
MNS	Microstimulateur neural sélectif
NLB	Neurostimulateur à lien bidirectionnel
NMOS	N-channel metal-oxide semiconductor
NSP	Neurostimulateur sélectif et permanent
PC	Personal computer
PCB	Printed circuit board
PIC	Programmable integrated controller
PMOS	P-channel metal-oxide semiconductor
PW	Pulse width
RAM	Random access memory
RF	Radio frequency
SG	Stimuli generator
SMA	Shape memory alloy
SWG	Stimuli waveform generator
VCO	Voltage controlled oscillator
VCUG	Voiding cystourethrogram

INTRODUCTION

Au cours des dernières décennies, l'électronique et la microélectronique ont vu un grand nombre de leurs applications dirigées vers le domaine biomédical. Celles-ci sont couramment utilisées pour le traitement des dysfonctions neurologiques ainsi que le remplacement ou l'assistance des organes assurant les fonctions vitales. Ces recherches ont engendré plusieurs types de dispositifs électroniques implantables. Les stimulateurs cardiaques (pacemakers) sont certainement les plus populaires, mais on peut aussi retrouver les stimulateurs musculaires pour les membres paralysés, les stimulateurs pour le rétablissement de l'audition (implants cochléaires), les implants urinaires ainsi que les implants dédiés à la stimulation du cortex visuel (stimulateurs corticaux).

Nous nous intéressons dans ce travail au stimulateur urinaire. Plusieurs tentatives ont été faites pour récupérer les fonctionnalités du système urinaire chez les patients paraplégiques, ceux atteints de maladies variées ou encore ceux ayant subies une lésion de la moelle épinière. Ces fonctionnalités sont souvent perdues due à une rupture de la communication entre les organes de l'appareil urinaire et les différents centres nerveux de la miction (moelle épinière et cortex cérébral); cette rupture étant causée par des lésions des voies sensorielles et motrices.

Les patients souffrant d'une lésion de la moelle épinière subissent une perte de contrôle volontaire de la vessie et peuvent avoir des contractions simultanées de la vessie et du sphincter (Dyssynergie) qui empêchent ou limitent l'évacuation complète de la vessie (Rétention). De plus, l'hyperactivité du muscle de la vessie (Detrusor) conduit à un

phénomène, appelé hyperreflexie de la vessie, qui est caractérisé par des contractions involontaires de celle-ci et des pertes urinaires.

Plusieurs techniques de traitement de ces dysfonctions urinaires, telles que la cathéterisation et les différentes chirurgies, ont été couramment utilisées et ont permis d'obtenir divers résultats. Cependant les risques d'infections dues aux cathéterisations répétitives et les désagréments irréversibles causés par les chirurgies ont accéléré le développement des techniques de stimulation électrique pour la récupération de ces fonctions vésicales.

Notre équipe de recherche PolySTIM a introduit au fil des années plusieurs versions de stimulateurs appliquant diverses techniques de stimulation. Les validations de ces différentes versions de stimulateur ont conduit au développement d'une nouvelle technique de stimulation dite sélective permettant la récupération du contrôle volontaire de la miction. De plus, l'utilisation d'une stimulation électrique de basse fréquence et de faible amplitude, appliquée aux racines sacrées contrôlant les organes du système urinaire, a permis de réduire l'hyperréflexie.

La dernière version de stimulateurs de PolySTIM a été la base des travaux présentés dans ce mémoire. Le système de stimulation existant a permis d'obtenir des résultats prometteurs; cependant, quelques déficiences lors de son utilisation, notamment au niveau du procédé d'isolation électrique de l'implant face au milieu biologique corrosif dans lequel il est utilisé, la robustesse des contrôleurs externes, la fiabilité des circuits critiques du stimulateur ainsi que sa dépendance à la tension d'alimentation

utilisée nous ont conduit à la réalisation d'une nouvelle version de stimulateur basée sur la même architecture mais corrigeant et améliorant les fonctionnalités déjà présentes.

Les pertes d'efficacité des stimulateurs lors des expériences de validation en phase chronique ont cependant rendu nécessaire la surveillance de l'évolution des implants ainsi que des paramètres caractérisant les interfaces électrodes-tissus pendant toute la durée d'une étude en phase chronique. Ces fonctions de mesure d'impédance de l'interface nerf-électrodes in-vivo permettent de prévenir et diagnostiquer les anomalies liées aux électrodes et aussi de préserver la sécurité des patients en cas d'apparition d'une anomalie quelconque (courant de stimulation élevé par exemple). Cette approche pourra permettre le développement de systèmes implantables beaucoup plus fiables car le retour de l'information a toujours été un des points faibles des stimulateurs et des systèmes implantables. Ils fonctionnent très souvent en boucle ouverte laissant l'usager dans l'ignorance complète de ce qui se passe réellement. De plus, les lacunes de la majorité des stimulateurs (manque de flexibilité des paramètres, limitation du nombre de canaux et des types de stimulation, manque de convivialité des interfaces ou encore manque de reprogrammabilité) ont motivé la conception et la réalisation d'une nouvelle version de stimulateur. Ce nouveau système intégré a été réalisé dans la technologie CMOS 0.18um pour parallèlement répondre à un besoin croissant de miniaturisation et de minimisation de la consommation d'énergie. Il combine ainsi toutes les fonctionnalités des systèmes de stimulation développés précédemment par notre équipe de recherche (Stimulation sélective et permanente, mesure d'impédance). De plus, un nouveau type de stimulation

dite flexible est intégré, et un deuxième canal indépendant de stimulation est ajouté. Ce dernier répond aux besoins de flexibilité et de reprogrammabilité précédemment énoncés.

Dans le chapitre 1 de ce mémoire, nous faisons une revue de littérature en présentant le système urinaire, les principes de la stimulation électrique neuronale ainsi que les systèmes de stimulation déjà existants. Le chapitre 2 présente un aperçu de l'architecture interne d'une nouvelle version de microstimulateur et décrit les modifications apportées à la version précédente du stimulateur. Dans le chapitre 3, nous présentons ce nouveau stimulateur intégré ainsi que son évaluation chez l'animal sous la forme d'un article soumis au journal « NEUROMODULATION », tandis que nous décrivons de façon détaillée la puce réalisée dans le chapitre 4. Finalement, nous dressons un bilan de travaux réalisés et présentons les résultats des systèmes testés dans le chapitre 5, et nous terminons par une série de recommandations sur la conception des futurs systèmes de stimulation et sur la méthodologie à suivre pour des futures expérimentations sur les dysfonctions du système urinaire.

CHAPITRE 1
STIMULATION ÉLECTRIQUE NEURALE ET SYSTÈMES EXISTANTS

1.1 Le système urinaire

1.1.1 Vue d'ensemble du système urinaire

Le système urinaire est composé de deux ensembles complémentaires: les appareils urinaires supérieur et inférieur.

Le premier est composé des calices rénaux, de bassinets, des uretères et a pour fonction de recueillir l'urine des reins et de la transporter jusqu'à la vessie. L'appareil urinaire inférieur (Figure 1-1) est formé de la vessie, de l'urètre, du sphincter et gère le stockage et la vidange volontaire de l'urine [2],[32].

La vessie est un réservoir d'urine dont les parois sont formées par un muscle appelé detrusor. Celui-ci a une structure lisse. L'urètre véhicule l'urine de la vessie au milieu extérieur. A la frontière vessie-urètre, on trouve un épaississement de la couche musculaire moyenne de la vessie mêlé à des fibres de l'urètre qui constitue le sphincter lisse. Le sphincter strié constitue le dernier élément de l'appareil urinaire inférieur et est composé de fibres musculaires striées qui entoure l'urètre.

Les expériences de Stewart et al tendent à démontrer qu'il existe une région spécifique de la moelle épinière appelée centre sacré de la miction, qui reçoit, intègre et

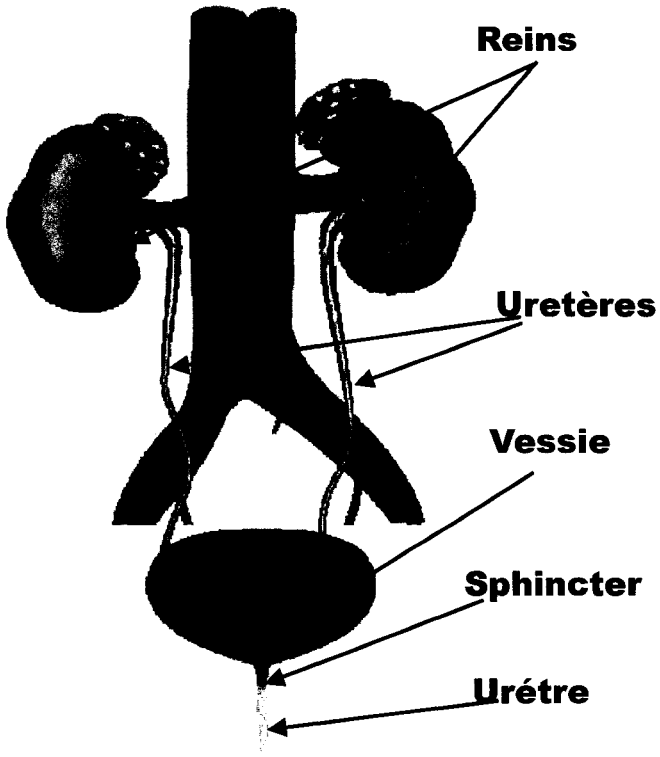


Figure 1-1: Vue d'ensemble du système urinaire.

transmet les influx nerveux moteurs et sensoriels ainsi que les activités inhibitrices qui influencent les réflexes de la miction. Le centre sacré de la miction est relié aux différents nerfs innervant l'appareil urinaire inférieur: les nerfs pelviens (detrusor), les nerfs sacrés S2 à S4 (detrusor et sphincter), ainsi que le nerf honteux (sphincter externe) [20]. Le detrusor et le sphincter externe partagent les nerfs sacrés comme source commune d'innervation. Ceux-ci contiennent un mélange de fibres somatiques et autonomes qui font circuler des influx nerveux dans les deux sens (afférent et efférent). Les fibres nerveuses afférentes sont sensorielles tandis que celles efférentes sont motrices. Les voies nerveuses autonomes afférentes (parasympathiques et sympathiques) stimulent le réflexe sacré de la miction tandis que les voies somatiques efférentes, contrôlées consciemment

par le cerveau (cortex cérébral) et le cervelet, sont responsables de la plupart des réflexes inhibiteurs tels que la contraction du sphincter externe[13]. L'harmonie du centre de la miction, du cortex cérébral et des différentes terminaisons nerveuses assure le bon fonctionnement des mécanismes du système urinaire; cela se traduit par une synchronisation entre les contractions et les relaxations respectives de la vessie et des sphincters.

1.1.2 Fonctionnement et dysfonctions urinaires

Le cycle normal de la miction est constitué d'une phase de remplissage de la vessie qui sera suivie par une phase de vidange de l'urine. Lors du stockage de l'urine, les fibres nerveuses sympathiques relâchent les fibres musculaires lisses de la vessie et maintiennent le sphincter lisse urétral fermé, permettant ainsi, grâce à une basse pression dans la vessie, le remplissage et l'accumulation d'un volume d'urine pouvant dépasser 500 ml. Lors de la phase mictionnelle, les fibres parasympathiques contractent la musculature lisse de la vessie et relâchent le sphincter urétral alors que le sphincter strié se contracte. Une augmentation de la pression dans la vessie opposée à une chute de la pression urétrale permet l'évacuation qui doit être complète [70].

Chez les patients paraplégiques, atteints de maladies variées (Parkinson, diabètes ou cancer) ou à la suite d'une lésion de la moelle épinière, apparaissent des dysfonctions du système urinaire. Ces dysfonctions sont liées à des lésions à un niveau plus ou moins élevé des voies efférentes ou afférentes causant ainsi une rupture de la communication entre les organes de l'appareil urinaire et les différents centres de la miction.

Suite à une lésion de la moelle épinière, une perte de contrôle volontaire de la vessie peut se produire et des contractions peuvent empêcher son évacuation complète. L'activité de la vessie est aussi associée à une augmentation de la résistance urétrale et de la pression vésicale qui limite l'évacuation chez la majorité des patients avec une lésion de la moelle épinière [71]. Les conséquences sont un grand volume résiduel d'urine ainsi que des reflux vésico-uréteraux, des développements bactériens, des infections, des défaillances rénales ou encore l'apparition d'incontinence [28].

L'incontinence quant à elle se caractérise par une impossibilité de la vessie de retenir adéquatement l'urine. Cela est due à une hyperactivité du detrusor qui conduit à des contractions involontaires de la vessie et à des pertes urinaires. Ce phénomène est couramment appelé hyperreflexie de la vessie [13].

Les techniques courantes de traitement de la rétention incluent différentes formes de cathéterisation (intermittente, continue,...). Souvent des narcotiques ou des chirurgies sont utilisés pour réduire la résistance urétrale. Les cathéterisations permettent des évacuations à basse pression de la vessie. Cependant les problèmes vésicaux persistent et seront accompagnés par des contractions dyssynergiques. De plus, certains patients peuvent développer des infections dues à ces cathéterisations répétitives [28].

Un autre axe de recherche exploré depuis de nombreuses années qui a fourni des résultats prometteurs et qui fait l'objet du présent mémoire est la stimulation électrique pour la récupération des fonctions vésicales.

1.2 La stimulation électrique fonctionnelle

La stimulation électrique fonctionnelle est décrite comme étant la stimulation d'un muscle privé de contrôle nerveux, dans le but de produire un mouvement utile fonctionnellement.

La stimulation électrique par l'intermédiaire des nerfs peut inhiber l'action des muscles. Aussi, elle peut fournir un moyen d'interagir avec les réflexes neuraux et influencer le comportement des organes innervés par les nerfs sacrés (vessie, sphincter, plancher pelvien) [13]. Le but de la stimulation électrique est de remédier aux dysfonctions de l'appareil urinaire sans cependant induire des dommages aux organes stimulés.

Les caractéristiques des muscles lisses et des muscles striés diffèrent aux niveaux des seuils de contractions, de la vitesse de conduction ou encore du seuil d'activation entre les axones somatiques et les fibres parasympathiques. Ces différences fournissent une alternative pour des manipulations electro-physiologiques du système urinaire neurologiquement déficient [66].

Il a été démontré que les stimulations à charges biphasiques balancées sont plus sécuritaires que celles à charges monophasiques et sont recommandées pour l'utilisation chronique [66].

1.2.1 Alternatives de stimulation

Quatre sites de stimulation peuvent être utilisés pour le contrôle de la vessie : le muscle de la vessie, les nerfs pelviens, les racines sacrées et la moelle épinière (Figure

1-2). Toutes ces méthodes de stimulation ont été évaluées dans des études expérimentales sur les animaux. Nous présentons ainsi un récapitulatif des avantages et des inconvénients de ces différentes techniques de stimulation [28],[31],[7],[25],[53].

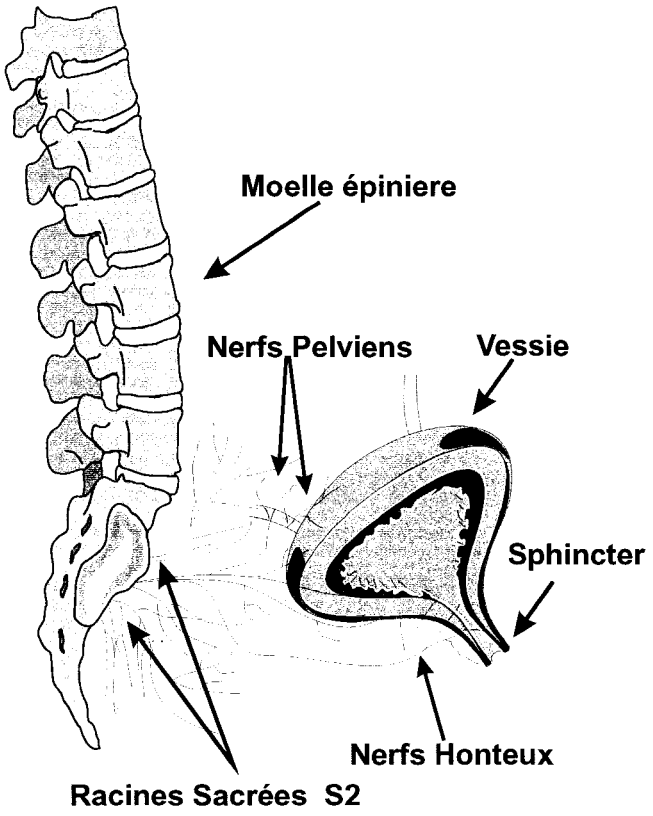


Figure 1-2 : Sites possibles de stimulation électrique pour évacuer l'urine de la vessie

a) Muscle de la vessie

Cette technique consiste à stimuler directement le muscle de la vessie (detrusor) par l'intermédiaire d'électrodes insérées dans ou sur la surface du muscle. Ce type de stimulation n'avait pas permis l'évacuation de la vessie car les contractions résultantes du detrusor n'étaient ni homogènes ni suffisantes. Aussi, la douleur associée à la stimulation

de la vessie chez certains patients, la co-activation des membres inférieurs et la diffusion du courant de stimulation dans des tissus avoisinants ont entraîné l'abandon de cette technique. Des progrès ont été réalisés dans la réduction de la diffusion du courant en raffinant le design des électrodes et le protocole de stimulation [69],[6], mais pas suffisamment cependant pour faire de la stimulation directe de la vessie une technique clinique viable [66]. Une étude de Merril en 1976 a indiqué que 52% des 64 cas de patients avec des lésions neuronales profondes ont empiré car l'implant n'arrivait pas à remédier à la dyssynergie du sphincter et du detrusor [8].

b) Moelle épinière

La stimulation du centre de la miction dans le cône médullaire a été examinée par Friedman et Nashold [43],[19], cependant cette technique n'a pas fait l'unanimité en raison de la faible sélectivité des électrodes de stimulation. Un placement précis des électrodes était nécessaire pour garantir un minimum de réussite. Cependant ici encore, la stimulation électrique utilisant des électrodes pénétrantes cause la double activation du muscle lisse de la vessie et du muscle strié du sphincter ainsi que des mouvements involontaires des membres inférieurs, de la transpiration, de l'hypertension ainsi que d'autres manifestations de la diffusion du courant [66].

c) Nerfs pelviens

La stimulation vésicale par le biais des nerfs pelviens affecte les nerfs honteux en induisant une excitation simultanée du sphincter et de la vessie (dyssynergie). Une

solution médicale a été proposée, soit la neurotomie des nerfs honteux. Ensuite, la stimulation du nerf S2 à proximité du site de la section a conduit à une évacuation à faible pression, avec une résistance négligeable du sphincter. Cependant l'approche de section de nerfs fait de cette technique une méthode peu engageante [28]. De plus, les expériences antérieures ont montré que la neurotomie des nerfs honteux est associée couramment à des difficultés d'érections chez les hommes paraplégiques [38].

d) Nerfs sacrés

La stimulation des racines sacrées antérieures est l'approche la plus utilisée en clinique. La stimulation intermittente des racines sacrées pour l'évacuation de la vessie chez les patients paraplégiques a été rapportée par Brindley [28],[10]. Lorsque les racines sacrées sont stimulées de façon appropriée, le sphincter et le detrusor se contractent en même temps, mais seule la musculature striée du sphincter aura le temps de se relâcher dans l'intervalle entre les trains de stimulation, permettant ainsi l'évacuation de l'urine [66],[69]. Cependant, cette méthode de stimulation intermittente est caractérisée par des pressions intra-vésicales élevées nocives pour les reins et par une évacuation par jets [46].

Une autre technique de stimulation se servant des nerfs sacrés est dédiée à la fatigue sphinctérienne par le biais des nerfs honteux. Ensuite, les nerfs sacrés sont stimulés pour évacuer l'urine. Ici, la stimulation électrique prend avantage des différences de durées des phases de contraction et de relâchement entre la vessie et le sphincter externe. Le concept de la fatigue a été appliqué pour éviter la section de nerfs (rhizotomies ou neurotomies). La différence au niveau des propriétés physiologiques

dans les muscles en réponse à la neurostimulation a justifié la technique de la fatigue sphinctérienne [38]. Cependant cette technique ne produit pas un réel relâchement du sphincter car les fibres striées à faible contraction constituant l'urètre sont très résistantes à la fatigue.

D'autres techniques réversibles, décrites comme blocages sélectifs, sont basées sur la différence dans les seuils d'excitation ou de blocage des fibres A-delta et A-alpha. Les différents types de blocages sont la collision des nerfs honteux [65], le blocage anodique par la stimulation des nerfs sacrés [9],[37] ou le blocage haute fréquence des nerfs honteux [33],[60].

Mortimer et al ont présenté une méthode d'activation sélective des petites fibres nerveuses dans les racines sacrées par la combinaison d'une excitation cathodique de toutes les fibres et un blocage anodique sélectif des grosses fibres [9], [1],[18]. Si à proximité d'une anode, les fibres nerveuses sont suffisamment hyperpolarisées par le courant anodique, les potentiels d'action ne peuvent plus se propager. Étant donné que les fibres à grand diamètre ont un seuil de courant inférieur pour leur blocage que celles à petit diamètre [1], l'activation sélective des petites fibres est possible en bloquant loin du site d'excitation (cathode) la propagation des potentiels d'action induits dans les grosses fibres (Figure 1-3). L'efficacité de cette méthode a été prouvée dans les nerfs périphériques [18]. Appliquée aux racines sacrées, l'activation sélective du muscle du detrusor est possible si les grosses fibres qui déclenchent le mécanisme de fermeture urétrale sont bloquées entraînant ainsi une nette amélioration de l'évacuation [46].

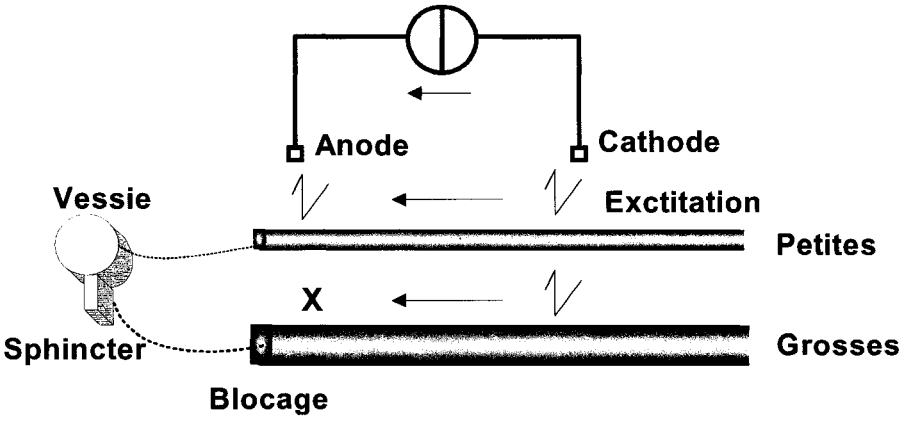


Figure 1-3: Principe du blocage anodique pour l'activation sélective des petites fibres.

En se basant sur le fait qu'une racine sacrée ventrale regroupe essentiellement deux classes de fibres, le blocage des grosses fibres somatiques peut être effectué par un stimulus de haute fréquence tandis que le detrusor peut (par le biais des petites fibres) être activé sélectivement par un signal de basse fréquence [75],[63]. Le blocage consiste à maintenir les nerfs et les extrémités motrices dans un état réfractaire empêchant ainsi les muscles de se contracter. Il a été montré [62] que la fréquence de blocage la plus efficace est de l'ordre de 600 Hz. Elle permet d'utiliser des durées d'impulsions moindres et de rester en dessous de la limite tolérable de charges par impulsion [60],[62],[77].

1.2.2 Incontinence

Les deux formes les plus communes de l'incontinence sont l'incontinence impérieuse (urgence mictionnelle) et l'incontinence de stress ou à l'effort. Le premier type d'incontinence se caractérise par une forte envie d'uriner qui résulte en une perte d'urine si le besoin n'est pas satisfait ou réprimé. Cette urgence est souvent reliée à l'instabilité du detrusor caractérisée par des contractions non inhibées de la vessie.

L'incontinence de stress est caractérisée par la perte involontaire d'urine lors d'activités physiques, lorsque l'on rie ou pendant un éternuement. Dans les cas plus sévères, la simple marche ou un changement de position peuvent déclencher l'évacuation d'urine. La cause principale est la dysfonction de l'urètre ou du col de la vessie qui conduit une insuffisance urétrale [13].

Walter et al ont introduit une méthode de traitement de l'incontinence par la neuromodulation pour induire une stabilité du muscle de la vessie. Celle ci consiste à stimuler les fibres somatiques afférentes dans les nerfs sacrés par un signal de basse fréquence assez effectif pour inhiber les activités de la vessie [71].

La plupart des systèmes implantables de stimulation fonctionnelle sont reliés aux nerfs par des électrodes dont le type et les formes varient selon les applications. Notre équipe de recherche a développé un nouveau type d'électrodes adapté à nos expérimentations [14]. La gaine de cette électrode se présente sous la forme de deux anneaux de platine fixés à l'intérieur d'un cylindre de silicone. Cette gaine s'enroule autour du nerf et les anneaux de platine assurent le contact électrique entre le nerf et le stimulateur. Le lien entre les deux contacts de la gaine et le microstimulateur est assuré par deux fils multibrins en acier inoxydable isolés par une couche de téflon. Pour maintenir l'électrode en place, nous utilisons des alliages à mémoire de forme que l'on intègre dans la structure de la gaine de silicone pour remplacer les fils de suture utilisés précédemment. Cette technique simplifie le travail du chirurgien, en lui évitant de faire des points de suture dans une cavité peu accessible lors de l'opération. De plus, cela ne risquera pas de comprimer le nerf à cause d'un point de suture trop serré. Pour permettre

la connexion directe des électrodes avec le microstimulateur, des connecteurs plaqués or et enrobés de silicone sont fixés aux extrémités des fils d'aciers inoxydables provenant des contacts de platine. Ainsi, seul le silicone sera en contact direct avec les tissus biologiques [54]. Ces précautions sont nécessaires car les dommages aux tissus sont liés aux propriétés mécaniques et chimiques des électrodes. Dans la région active des électrodes, à l'endroit où le métal est en contact direct avec les tissus, des dommages peuvent survenir en raison des réactions électrochimiques à la surface de l'électrode durant la stimulation [11],[12],[17],[42].

1.3 Dispositifs et systèmes existants

Nous avons survolé les bases de la stimulation électrique pour la réhabilitation du système urinaire ainsi que les différents principes et techniques de stimulation. Au niveau des systèmes implantables, plusieurs dispositifs ont été conçus et rendus commercialement disponibles et aident à améliorer le confort des patients dans leur vie sociale. Nous présentons une brève rétrospective de ces systèmes ainsi que leurs principales caractéristiques.

1.3.1 Systèmes commerciaux

Tout d'abord les laboratoires Avery® ont présenté en 1974 un stimulateur pour le traitement de la rétention. La première version effectuait une stimulation en tension monophasique; une version bipashique a été ensuite rendue disponible. En 1977 l'adaptation d'un pacemaker par Gorgis® conduit à un système ajustable à un canal pour

la rétention. La particularité de ce système est l'utilisation d'un aimant pour la gestion de l'alimentation. Finetech® a présenté en 1985 un triple stimulateur transcutané générant des stimuli monophasiques. En 1987, Phisico-Med® a introduit sa version du stimulateur mono-canal pour la rétention. Finalement la compagnie Medtronic [41] a présenté en 1997 un stimulateur transcutané qui traite l'incontinence. Plusieurs versions ont suivi et de nos jours, leur thérapie InterStim® permet de traiter l'incontinence et la rétention par le biais d'une stimulation électrique sur les nerfs sacrés.

Parallèlement à cela, plusieurs équipes de recherche ont proposé différentes prothèses vésicales. Celles ci ont connu plus ou moins de succès; Le manque d'engouement étant le plus souvent du à certaines lacunes de ses stimulateurs, notamment au niveau du manque de flexibilité des paramètres, des limitations au niveau du nombre de canaux et des types de stimulation (mono versus biphasique), de l'absence le plus souvent d'une interface usager conviviale ou encore de la faible efficacité de transmission de données et d'énergie. En fait, peu des stimulateurs disponibles combinaient la totalité de ces paramètres. C'est ainsi que la recherche a été de plus en plus axée vers la conception de systèmes implantables transcutanés complètement reprogrammables. Nous allons nous limiter à un survol des différents systèmes qui ont conduit à l'élaboration du système présenté dans ce mémoire.

1.3.2 Systèmes intégrés

Sawan et al présentent tout d'abord en 1992 un système multicanal implantable [50]. Ce système intégré et réalisé dans la technologie CMOS 3um est en mesure de

générer des stimuli par l'intermédiaire de 8 canaux monopolaires (4 canaux bipolaires). Aussi, deux types de contrôleurs ont été présentés pour faire fonctionner ce système. Le premier est basé sur un micro-ordinateur avec un logiciel dédié, tandis que le deuxième regroupe un circuit intégré dédié dans un module portable. Les deux contrôleurs envoient les paramètres de stimulation ainsi que l'énergie nécessaire par l'intermédiaire d'un lien inductif avec une porteuse à 20 MHz.

Arabi et Sawan ont ensuite introduit un micro-stimulateur intégré implantable [3], [4]. Avec huit canaux monopolaires transformables en 4 canaux bipolaires, il présente un grand degré de reprogrammabilité ainsi qu'une flexibilité permettant de l'adapter facilement. Capable de générer une grande variété de formes d'ondes, il peut combiner jusqu'à quatre différentes fréquences programmables sur chaque onde. Pour la fiabilité de la transmission, un protocole de détection et de correction d'erreur est utilisé. Une approche basée sur les techniques de chaînes de balayage est aussi utilisée pour améliorer la testabilité structurelle. Un circuit intégré dédié (ASIC) a été réalisé dans la technologie 1.2um CMOS4S. Ce système était dédié à la récupération des deux fonctions urinaires (rétention et évacuation). Un canal dédié alimenté en permanence par une batterie contrôle le sphincter pour éviter l'incontinence en appliquant une stimulation basse fréquence en continu sur le nerf. Enfin l'une des dernières particularités réside dans le fait que les paramètres de stimulation sont stockés dans des cellules de RAM. Celles-ci contiennent des échantillons des demies périodes de chaque forme d'onde primaire. La lecture séquentielle de ces cellules permet de générer les formes d'ondes programmées.

1.3.3 Stimulation sélective

Une nouvelle technique de stimulation, dite stimulation sélective, basée sur le blocage haute fréquence des activités sphinctériennes par le biais des nerfs sacrés S2 est ensuite introduite par Robin et al [48],[47]. Ces derniers présentent un nouveau système implantable pour la stimulation sélective. Celui-ci contrôle un seul canal de sortie analogique servant à générer des stimuli de courant bipolaires à deux fréquences indépendantes et peut être commandé à l'aide de deux types de contrôleurs : un premier basé sur une interface PC pour les médecins avec un logiciel dédié permettant une flexibilité accrue pour l'ajustement des paramètres et un contrôleur portable dont l'architecture repose sur la combinaison d'un FPGA et d'une EPROM pour sauvegarder plusieurs combinaisons de paramètres (maximum huit). Le Tableau 1.1 présente les six paramètres contrôlables.

Tableau 1.1: Description des paramètres de la stimulation sélective

Abréviations	Description du paramètres*
LFP	Période séparant 2 stimuli à basse fréquence
HFP	Période séparant 2 stimuli à haute fréquence
LFW	Largeur des stimuli basse fréquence
HFW	Largeur des stimuli haute fréquence
LFA	Amplitude des stimuli basse fréquence
HFA	Amplitude des stimuli haute fréquence

*Les plages de variations sont présentées au chapitre 2

La Figure 1-4 représente la forme d'onde typique du signal de stimulation sélective généré par le microstimulateur, d'où son nom de microstimulateur neural sélectif (MNS).

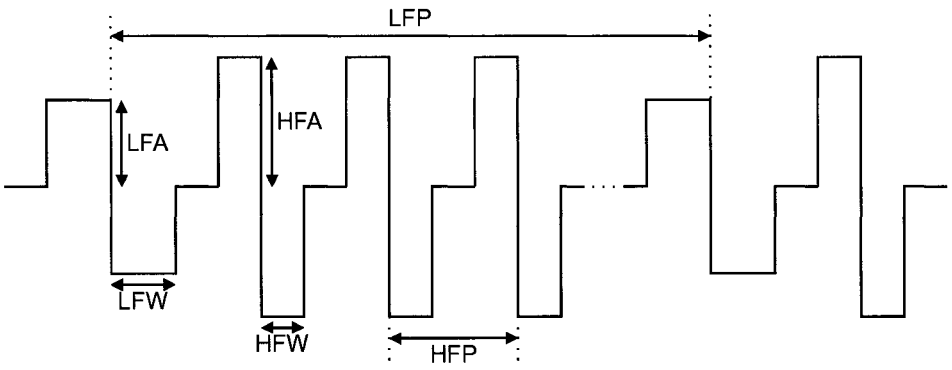


Figure 1-4: Forme d'onde d'un signal de stimulation sélective

L'implant lui même est monté sur un circuit imprimé circulaire de 4 centimètres de diamètre et est structuré autour d'un FPGA qui contient toute la logique de contrôle de génération de stimuli. On retrouve sur l'implant un démodulateur AM ainsi qu'un bloc de récupération de données. Celui-ci permet de retrouver les données ainsi que l'horloge initialement codés ensemble en format Manchester. Un convertisseur numérique à analogique (CNA) génère une tension proportionnelle à l'amplitude de stimulation souhaitée. Celle-ci est ensuite convertie en courant par un amplificateur opérationnel configuré en source de courant. Le courant est finalement injecté dans le nerf via une structure de commutateurs en H, pour assurer la bipolarité des stimuli. Le FPGA par l'intermédiaire de deux générateurs de formes d'ondes produit les commandes pour les stimuli. Ceux-ci sont composés de deux trains d'impulsions bipolaires caractérisées par leur amplitude, leur fréquence ainsi que leur largeur. Les formes d'ondes sont ensuite

superposées pour produire des stimuli équilibrés et balancés pour éviter la polarisation du nerf par l'injection et l'accumulation de charges au niveau de l'interface électrode-tissu. Le système complet a été testé in vivo dans des expérimentations aussi bien en phase aiguë que chronique. Les résultats obtenus après plus de cinq mois d'évaluation ont présenté une augmentation supérieure à 50% dans le volume moyen d'urine évacué ainsi qu'une réduction du volume résiduel jusqu'à 9%.

Ces résultats ont encouragé l'intégration du système qui a été effectuée en utilisant la technologie BICMOS 0.8um.

Les tests en phase chronique ont cependant montré une perte d'efficacité de stimulation ou encore une interruption soudaine du fonctionnement de l'implant. Ces défaillances ont amené l'équipe PolySTIM à développer un système qui permettait de surveiller le fonctionnement de l'implant ainsi que l'évolution de certains paramètres [54] caractérisant les interfaces électrodes-tissus pendant toute la durée d'une étude en phase chronique.

Schneider et al ont conçu un nouveau système implantable capable de mesurer l'impédance de l'interface nerf-électrodes in-vivo et de retourner la valeur mesurée vers un afficheur externe [54]. Ce neurostimulateur à lien bidirectionnel (NLB) intègre aussi la fonction de stimulation dite sélective, décrite plus haut, permettant la miction volontaire. Des tests en laboratoire ont démontré la fonctionnalité souhaitée du système.

Le désir de trouver une solution à l'hyperreflexie et d'intégrer cette solution aux systèmes précédents a motivé ensuite le développement d'une troisième génération de neurostimulateurs qui inclut une nouvelle technique de stimulation, appelée stimulation

permanente, assurant une réhabilitation complète du système urinaire (miction complète et réduction ou annulation de l'hyperreflexie). Ce neurostimulateur sélectif et permanent (NSP) effectue deux sortes de stimulation; sélective et permanente [54]. La stimulation permanente est une stimulation à très basse fréquence et très basse amplitude qui nécessite une alimentation par pile. Les paramètres de ce type de stimulation sont présentés dans le Tableau 1.2 .

Tableau 1.2: Description des paramètres de la stimulation selective

Abréviations	Descriptions du paramètre
Amp	Amplitude des stimuli
Freq	Fréquence des impulsions
PW	Durée des pulses
Ton	Temps d'activation de la stimulation
Toff	Temps d'inactivation de la stimulation

*Les plages de variations sont présentées au chapitre 2

Ce stimulateur possède ainsi deux sources d'alimentation qui sont adéquatement gérées pour maximiser la durée de vie de la pile. La présence d'une mémoire non volatile ainsi que la fonction d'économie d'énergie du microcontrôleur utilisé, associées aux bons résultats obtenus en stimulation sélective avec les FPGA dans les versions précédentes ont conduit à une architecture mixte utilisant deux types de contrôleurs : un FPGA alimenté par l'énergie inductive pour réaliser la fonction de stimulation sélective et un microcontrôleur (PIC) alimenté par une pile dédiée à la stimulation permanente. Les 2 contrôleurs partagent le même étage de sortie.

Le FPGA est alimenté à partir du lien inductif RF par une antenne et effectue la récupération ainsi que le décodage de l'ensemble des données (paramètres nécessaires à la génération des stimuli) transmises par le contrôleur externe à l'implant. Le FPGA est aussi responsable du transfert de données au PIC qui est uniquement alimenté par la pile. Dans le cas d'une stimulation sélective, le FPGA prend le contrôle de l'étage de sortie qui est alors alimenté par l'énergie RF. Lors d'une stimulation permanente, l'implant est alimenté par la batterie et le PIC utilise des paramètres de stimulation préprogrammés.

Le contrôleur externe envoie à l'implant un signal RF de 20MHz modulé en amplitude à un taux de 300Kbps. Un circuit analogique de régulation et de démodulation permet l'alimentation en énergie de la majeure partie de l'implant ainsi que la récupération d'un signal codé Manchester. De ce signal, le FPGA reconstitue l'horloge (300KHz) et extrait les paramètres de stimulation.

L'étage analogique de réception responsable de la régulation de tension et du décodage des données est identique aux circuits utilisés dans les versions antérieures du stimulateur (MNS et NLB). Le signal RF est redressé à double alternance puis filtré pour ensuite être régulé par une diode Zener à seuil ajustable. La démodulation est effectuée par un détecteur d'enveloppe.

Un prototype du neurostimulateur a été réalisé sur un circuit imprimé avec des circuits intégrés commerciaux programmables et des composants discrets. Huit exemplaires ont fait l'objet d'une étude expérimentale à long terme sur des animaux de laboratoire, afin de caractériser la fiabilité et la fonctionnalité de l'implant [54]. Les résultats de cette étude ont montré que le système de stimulation proposé ainsi que la

technique chirurgicale exercée répondaient aux attentes quant à la résolution du problème d'évacuation et permettaient la réduction des manifestations de l'hyperréflexie chez les animaux paraplégiques.

1.4 Conclusion

Nous avons présenté dans ce chapitre une vue d'ensemble du système urinaire et énoncé brièvement les différentes dysfonctions qui peuvent survenir lors de son fonctionnement. Les différentes techniques de stimulation électrique pour la correction de ces dysfonctions ont été aussi exposées. Nous avons poursuivi par la présentation de différents systèmes commerciaux et expérimentaux appliqués à la réhabilitation du système urinaire. Nous avons pu nous rendre compte que la stimulation sélective des nerfs sacrés basée sur la technique de blocage haute fréquence semblait être prometteuse pour la récupération de la miction volontaire et que l'application d'une stimulation permanente à basse fréquence permet la réduction de l'hypperreflexie chez les paraplégiques. Nous allons présenter dans le prochain chapitre les détails du design d'une nouvelle version du stimulateur exploitant les avantages présentées par les techniques de stimulation présentées ci haut.

CHAPITRE 2
SYSTÈME DE STIMULATIONS SÉLECTIVE ET PERMANENTE

2.1 Introduction

Notre équipe de recherche a développé lors des dernières années un système de stimulation électrique fonctionnelle effectuant deux types de stimulation : une dite sélective pour permettre de récupérer la fonction de miction volontaire chez les patients souffrant de lésions de la moelle épinière et une autre, dite permanente qui a pour but de réduire, et à long terme supprimer les symptômes de l'hyperreflexie [54]. Ce neurostimulateur sélectif et permanent a fait l'objet d'expérimentations chroniques chez des chiens à l'animalerie de l'Université McGill. Ces expérimentations, étalées généralement sur des durées moyennes de huit mois, ont permis de démontrer globalement le fonctionnement du système. Notamment plus de 75% du volume d'urine a été évacué lors des stimulations sélectives et une très nette réduction de la phase d'hyperréflexie a pu être observée sur tous les sujets traités par une stimulation permanente. Ce stimulateur a cependant présenté certaines défaillances lors de son fonctionnement. Celles-ci ont pu être parfois attribuées à des bris aux niveaux de contrôleurs externes, des électrodes de stimulation et quelques défauts ont aussi été notés au niveau de l'implant:

- L'isolation électrique insuffisante de la pile entraînait cette dernière à se décharger rapidement ce qui réduit la durée de fonctionnement du microstimulateur implanté;
- Un arrêt définitif de la stimulation sélective survient lorsque la pile est déchargée. Cela est entraîné par la mise en commun des circuits de stimulations sélective et permanente au niveau de l'étage de sortie;
- Une saturation de l'amplificateur opérationnel utilisé comme source de courant apparaît lorsque l'impédance du nerf devient trop élevée.

Pour éliminer ces défauts, nous avons donc entrepris le design d'une nouvelle version de ce système. Ceci représente l'objet de ce chapitre.

2.2 Spécifications du système initial

Le neurostimulateur projeté sera comme son prédécesseur, le NSP, dédié au rétablissement de la miction volontaire et à la réduction de l'hyperreflexie. Son architecture s'articule sur la combinaison des deux contrôleurs intégrés (PIC et FPGA) fonctionnant de façon mutuellement exclusive, excepté lors de la communication entre ces deux contrôleurs. Chacun d'eux est responsable d'un mode de stimulation et produit les signaux nécessaires à la génération des formes d'ondes souhaitées. Un contrôleur de bus est présent dans l'architecture afin de pouvoir mettre les sorties du FPGA en haute impédance lorsque celui-ci n'est pas alimenté; propriété que ne possède pas le FPGA que nous utilisons. Cette précaution est nécessaire car dans l'architecture du précédent

stimulateur, le FPGA et le PIC se partagent les différents bus de données nécessaires à la génération des stimuli. Lors de la stimulation permanente, le contrôleur de bus est désactivé et permet ainsi d'éviter que les signaux générés par le PIC ne viennent, par l'intermédiaire des ports d'entrées/sorties du FPGA, induire une tension d'alimentation. Les deux types de stimulation cités ci-dessus feront l'objet des prochaines sections.

2.2.1 Stimulation sélective

La stimulation sélective est dédiée à la réduction de la dyssynergie entre le sphincter et la vessie. Cette technique de stimulation permettant l'évacuation de l'urine nécessite deux types d'ondes qui sont superposées pour produire la série de stimuli requis. Un signal à haute fréquence effectue le blocage des grosses fibres somatiques innervant le sphincter tandis que les petites fibres parasympathétiques du muscle de la vessie sont activées sélectivement par un signal en mode courant à basse fréquence.

La forme d'onde typique d'un signal de stimulation sélective ainsi que les paramètres qui s'y rattachent ont été présentés au chapitre 1. Le Tableau 2.1 contient la description de ces paramètres, leurs valeurs limites utiles ainsi que la résolution autour de leur valeur typique.

2.2.2 Stimulation permanente

Pour le traitement de l'hyperreflexie, le stimulateur génère de façon autonome et continue un signal de stimulation permanente (Figure 2-1). Ce signal, de basse fréquence, est un courant bipolaire, de faible amplitude, et forme un train d'impulsions avec des temps d'activation et d'inactivation programmables.

Tableau 2.1: Paramètres de la stimulation sélective

	Basse fréquence			Haute fréquence		
Paramètre	Amplitude	Fréquence (1/Période)	Largeur d'impulsion	Amplitude	Fréquence (1/Période)	Largeur d'impulsion
Acronyme	LFA	LFP	LFW	HFA	HFP	HFW
Unité	mA	Hz	µs	mA	Hz	µs
Plage	0 – 2	18 – 2000	3.3 – 210	0 - 2	290 - 2000	3.3 - 210
Résolution	8 µA	3 mHz	3.3 µs	8 µA	1.2 Hz	3.3 µs
Valeur typique	0.9	30	175	1.3	600	100

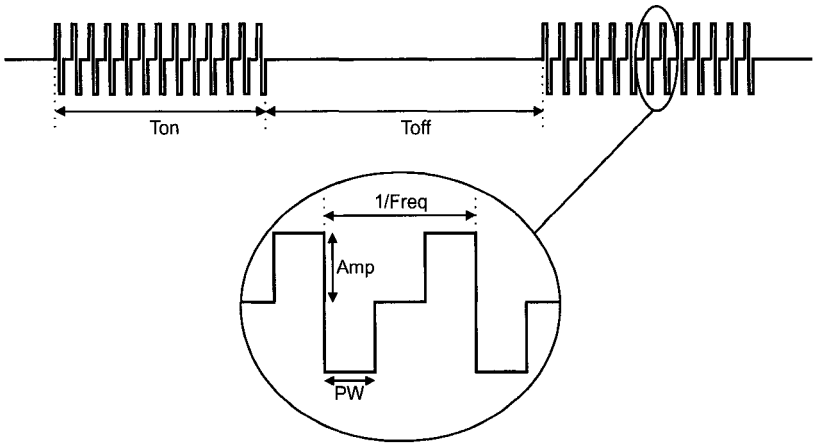


Figure 2-1: Forme d'ondes d'un signal de stimulation permanente

Le Tableau 2.2 décrit les 5 paramètres programmables utilisés pour générer les formes d'ondes de stimulation permanente et présente leurs valeurs de plage et de résolution.

Tableau 2.2: Paramètres pour la stimulation permanente

Paramètre	**Amplitude**	**Fréquence (1/période)**	**Largeur d'impulsion**	**Durée active**	**Durée inactive**
Acronyme	Amp	Freq	PW	Ton	Toff
Unité	µA	Hz	µs	sec	sec
Plage	0 - 1000	5 - 100	10 - 300	10 - 60	1 - 10
Résolution	10	1	10	5	1
Valeur typique	200	30	175	20	10

2.3 Modifications du design

L'architecture du stimulateur proposé est inspirée des travaux passés de notre équipe PolySTIM [54]. Nous présentons dans les prochaines sections les modifications pertinentes et les nouveautés apportées au design.

2.3.1 Passage de 3.3V à 5V

La tension d'alimentation du précédent système était fixée à 3.3V, qu'elle provienne de la batterie ou du circuit de régulation de l'énergie RF. Pendant que les nouvelles technologies réduisent les tensions d'alimentation pour réduire la consommation d'énergie, nos applications nécessitent de maintenir, voire augmenter ces tensions. En effet, après quelques périodes de stimulation, il a été observé que l'impédance du nerf avait tendance à augmenter et pouvait doubler voir tripler sa valeur nominale (environ 1KΩ). Ces variations d'impédance entraînaient ainsi une saturation en tension de l'amplificateur opérationnel de sortie. En effet, pour maintenir un courant constant dans une résistance de plus en plus élevée, l'amplificateur opérationnel doit augmenter la tension aux bornes de la charge jusqu'à causer une saturation lorsque cette tension atteint la tension d'alimentation.

Sur le schéma de la Figure 2-2, nous pouvons voir que le courant de stimulation est égal à $I_{nerf} = V_i / R_i$. Dans le système initial, la tension maximale d'entrée Vi était égale à 1.22V. De ce fait, la résistance d'entrée Ri avait été choisie égale à 610Ω afin de limiter le courant maximal dans le nerf à 2mA. Si on considère la boucle de sortie, nous obtenons l'équation (1) :

$$Rnerf = \frac{Vout - Vi}{Inerf} \qquad (2.1)$$

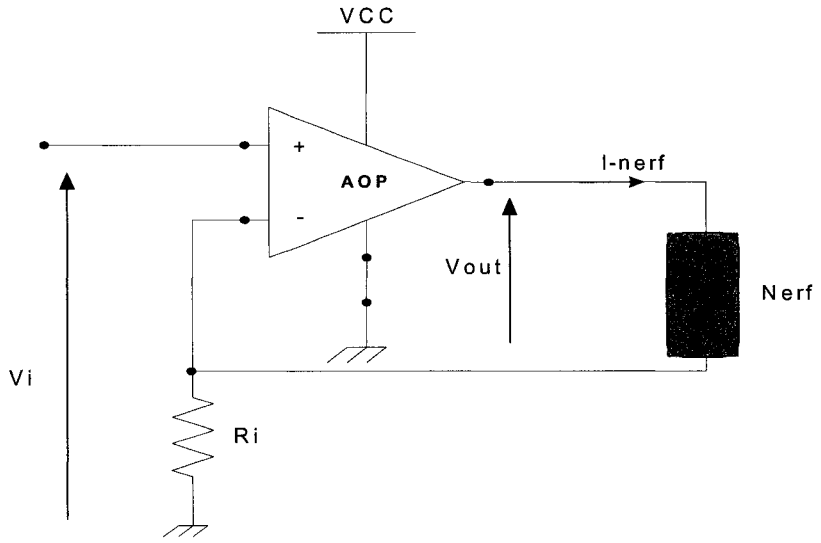


Figure 2-2: Schéma équivalent de la source de courant du système

En utilisant les valeurs maximales des paramètres et avec la tension de sortie maximale $V_{out}max$ = Vcc = 3.3V, nous obtenons la plage de variation de l'impédance du nerf ($0 < R_{nerf} < 1.04\ K\Omega$) pour un courant maximum de 2mA.

Pour accommoder l'augmentation de l'impédance, nous avons ainsi décidé de monter la tension d'alimentation à 5V. Des recherches ont été effectuées pour trouver une pile de format appropriée (*coin-type*) fournissant une tension de 5V. Le but était de réduire la complexité du design et passer tout le système à une tension d'alimentation unique de 5V, mais la tension maximale fournie par les piles de ce format était 3.3V. Nous nous sommes cependant rendus compte que ce problème ne survenait que lors des stimulations sélectives car la stimulation permanente requiert des signaux de faible

amplitude et nécessite peu de courant. De ce fait, la modification ne s'est appliquée qu'au circuit d'alimentation de la partie sélective.

La régulation de tension de l'énergie envoyée de l'extérieur est effectuée par un composant électronique qui est basée sur une diode Zener (ZR431) de précision ajustable.

La Figure 2-3 montre le circuit du ZR431. Il est indiqué que la tension de sortie V_{out} est ajustable en fonction de V_{ref} (équation 2.2). À titre d'exemple, une tension de référence V_{ref} = 2.5V (spécifications du composant) et deux résistances R1 et R2 identiques ont permis de fournir le 5V souhaité. Nous avons ensuite procédé à la vérification des tensions d'alimentation de tous les composants électriques qui doivent supporter les deux alimentations requises.

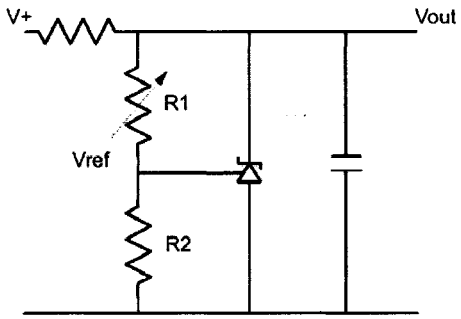


Figure 2-3: schéma de fonctionnement du ZR431

$$V_{out} = \left(1 + \frac{R_1}{R_2}\right) x V_{ref} \qquad \textbf{(2.2)}$$

2.3.2 Étage de sortie amélioré

Après avoir changé à 5V la tension d'alimentation du bloc effectuant la stimulation sélective, nous avons décidé de dissocier au maximum les deux circuits de stimulation. Cette séparation se justifie par le fait que l'un des plus gros défi rencontré

par les concepteurs de circuits et systèmes électroniques est la cohabitation de plusieurs alimentations à différentes tensions; situation à laquelle nous faisons face dans notre design. De plus, l'un des problèmes rencontrés lors des expériences chroniques avait été l'interruption du fonctionnement complet du système après que la pile soit déchargée, cela dû au fait que les deux contrôleurs se partagent l'étage de sortie et que celui-ci était alimenté par la pile.

Dans ce nouveau design, nous avons opté de doubler les blocs formant l'étage de sortie. Cet étage regroupe les convertisseurs numérique/analogique et sources de courant (Figure 2-4). Cette opération réduit à deux les liens de communication entre les deux circuits de stimulation : le premier concerne le dialogue entre les deux contrôleurs, et le second se trouve au niveau de la liaison avec le nerf. La communication entre le FPGA et le PIC est nécessaire car, rappelons le, c'est le FPGA qui transmet au PIC le mode et les paramètres de stimulation. Quand à la liaison avec le nerf, le commutateur de courant est aussi bien fonctionnel à 3.3V qu'à 5V et vu qu'il est alimenté en permanence, il ne peut avoir d'interférences de tension à ce niveau. Les deux amplificateurs de sortie sont reliés entre eux et alimentent le nerf. Dans n'importe quel cas des types de stimulation, un seul de ces amplificateurs se trouve alimenté et activé à la fois.

2.3.3 Communication PIC-FPGA

Tel qu'il a été décrit dans la section précédente, les deux circuits de stimulation communiquent au niveau des deux contrôleurs (PIC et FPGA) lors de la transmission des données de stimulation. Rappelons les principes de cette communication :

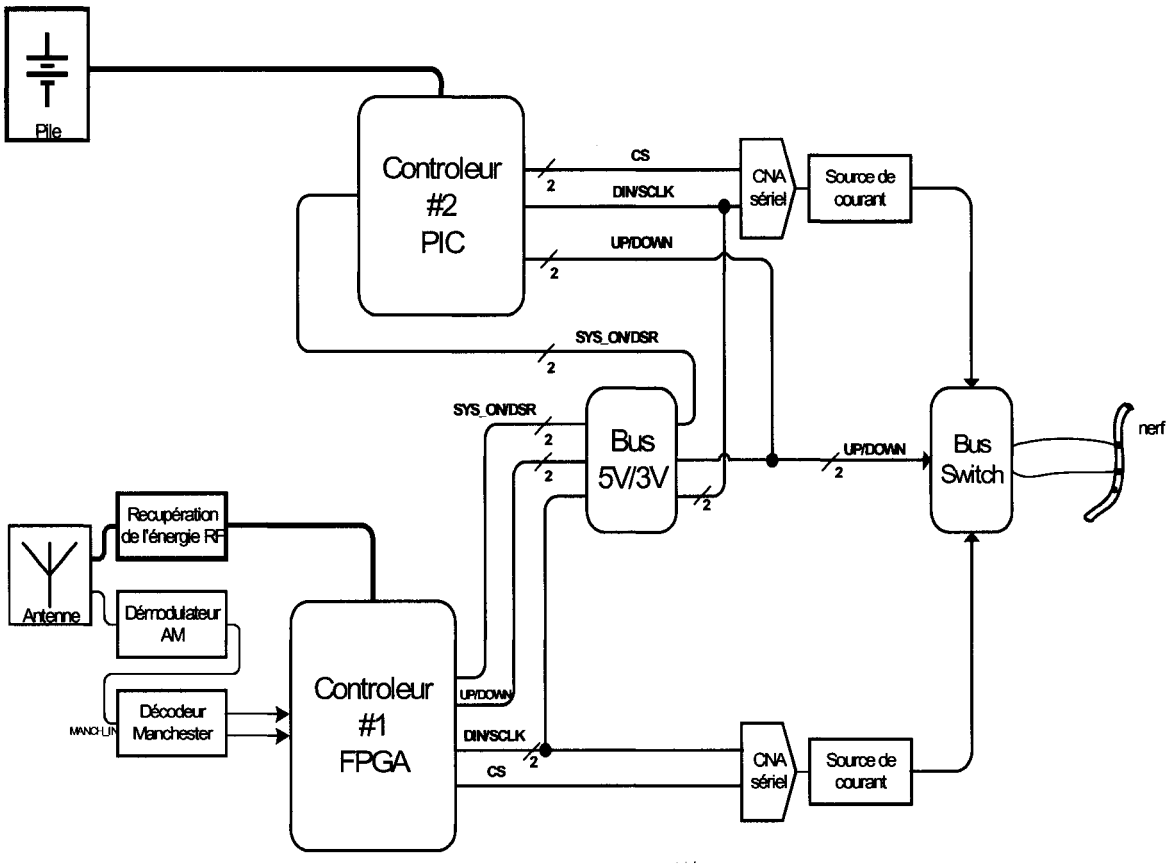


Figure 2-4 : Diagramme bloc du nouveau stimulateur implantable.

- Le FPGA décode tout d'abord les données reçues en format manchester. Dans le cas d'une stimulation sélective, il avise le PIC en mettant un niveau logique '1' sur la ligne SYS_ON et un niveau logique '0' sur la ligne DSR (Figure 2-4). Le PIC met alors ses sorties en mode haute impédance et envoie un signal au FPGA par l'intermédiaire de la ligne READY ; le FPGA procède ainsi à préparer et générer les stimuli.
- Après le décodage des données et l'identification du mode de stimulation permanente par le FPGA, celui-ci prévient le PIC en mettant un niveau logique '1' sur les lignes DSR et SYS_ON. Le PIC se met en mode récupération de données et transmet un

signal de synchronisation en mettant un niveau logique '1' sur la ligne READY. Le FPGA transmet alors les paramètres de stimulation au PIC par la ligne DIN à une vitesse fixée par ce dernier qui a pris le contrôle de l'horloge du FPGA par la ligne SCLK. Après réception et vérification de la validité des paramètres, le PIC déclenche la création des stimuli et leur génération dès l'arrêt de la transmission de l'énergie RF.

Après le changement d'alimentation du circuit de stimulation sélective à 5V, nous nous sommes retrouvés face à un problème de communication dans la mesure où, le FPGA, avec ses entrées et sorties à 5V, échange des données avec le PIC qui lui fonctionne encore à 3.3V.

Nous avons tout d'abord vérifié si le PIC en étant alimenté à 3.3V pouvait recevoir des données à 5V ; cependant l'architecture interne du microcontrôleur, bien que les broches d'entrées/sorties soient équipées de diodes de protection, ne permettait pas ce mode de fonctionnement. Comme solution alternative, nous avons utilisé des résistances montées en pont diviseur de tension sur les différentes lignes impliquées. Cette solution supposait l'utilisation de grandes valeurs de résistances pour minimiser la consommation de courant. De plus, certaines lignes du PIC, utiles à la communication (DIN_PIC, S_CLK), étaient utilisées en mode de stimulation permanente et ne devaient pas alimenter le FPGA. Devant cette contrainte, nous avons décidé de remplacer le contrôleur de bus initial (SN74CBT3244) par un autre contrôleur (SN74CBTD3384PW) ayant les mêmes caractéristiques que le précédent en plus de la conversion de niveau (*Level Shifting).* De cette façon, les entrées du contrôleur de bus sont à 5V tandis que ses sorties sont converties à 3.3V. Ce nouveau contrôleur de bus possède deux paires

d'entrées/sorties supplémentaires par rapport au précédent et sa tension d'alimentation est à 5V. Toutes les lignes de communication entre les deux contrôleurs passent via le contrôleur de bus. Vu que les lignes DIN_FPGA et SCLK_FPGA du FPGA servent aussi à la génération de stimuli pour les deux modes de stimulation, nous les avons doublées et adaptées leur voltages pour éviter tout conflit de tension.

Après ces transformations matérielles, nous avons effectué les modifications logicielles en conséquence dans les programmes du PIC et la configuration du FPGA. En effet dans la version antérieure, les lignes DSR et SYS_ON étaient en liaison directe entre le FPGA et le PIC. Ces connections lui permettait ainsi de détecter immédiatement tout changement de niveau sur ces signaux et par conséquent un changement de mode de fonctionnement.

Dans le nouveau design, ces deux lignes sont interrompues par le contrôleur de bus. De ce fait, lorsque celui-ci n'est pas activé, le PIC ne peut pas détecter le mode de fonctionnement de l'implant. L'activation du contrôleur de bus était précédemment contrôlée par une combinaison du signal de synchronisation READY du PIC et du signal de commande SYS_ON du FPGA. Lorsque ces deux signaux étaient activés en même temps (niveau logique haut), par l'intermédiaire d'une porte NAND ils activent le signal Z_CTRL (actif bas) qui contrôle les entrées « *Output Enable* » du composant. Tel qu'il a été expliqué plus haut, le PIC n'active le signal READY qu'après avoir détecté le mode de fonctionnement. Nous devions ainsi modifier les programmes afin que le bus soit activé dès la détection du mode par le FPGA. Cette modification pouvait être effectuée au niveau du FPGA en supprimant la dépendance du signal Z_CTRL au signal READY ou

au niveau du PIC en initialisant le signal READY avec un niveau logique '1'. Nous avons choisi de modifier le programme du PIC, afin d'assurer que le PIC était complètement initialisé avant le début de la génération de stimuli par l'implant. De plus, toute modification ultérieure était facilitée par la fonction de programmation *On-circuit* dont dispose ce composant. Les nouvelles versions des programmes modifiés sont disponibles à l'annexe A.

2.4 Encapsulation des implants

Nous avons présenté les défaillances de l'implant au niveau de l'électronique; cependant celles-ci étaient en partie causées par la durée de vie raccourcie de la pile encapsulée. Les conclusions avaient été que la pile se déchargeait beaucoup plus vite que prévu due à la pénétration des fluides corporels à l'intérieur de l'implant et cela malgré la couche protectrice de silicone biocompatible (silastic) qui recouvrait l'implant. Ces fluides engendraient le passage d'un certain courant de fuite indésirable entre les pôles de la pile et en accéléraient la décharge.

Une solution proposée pour remédier à cette situation était de remplacer le silicone par une époxy à deux phases pour encapsulation permettant une meilleure protection de la pile. Nous nous sommes ainsi lancés dans la recherche d'une époxy « bi-phasique » dont les principaux critères étaient la biocompatibilité pour éviter toute forme de rejet par l'organisme après implantation, ainsi que la simplicité de son application pour encapsuler l'implant.

Deux échantillons d'époxy ont été considérés : le premier est une époxy fournie par la compagnie Miller Polymer Products INC. Celle-ci (STYCAST 1269) est constituée de deux catégories, l'une transparente et liquide tandis que la deuxième (le solidifiant) est opaque et pâteuse. Ces deux catégories doivent être mélangées dans un rapport de un pour un. Le deuxième échantillon d'époxy est fourni par la compagnie Epoxy technologie (EPO-TEK®) et est composé aussi des deux catégories de liquide, mais elles doivent être dans ce cas mélangées dans un rapport de quatre pour un.

Les deux époxies ont été soumis à des caractérisations en laboratoire, notamment au niveau de l'infiltration, la solidité ainsi que la facilité de traitement. Dans l'ensemble, les deux répondaient bien à nos exigences. Il faut noter que le résultat final produit par les deux types est un produit rigide très solide. Cela donc implique que les encapsulations d'implants avec ces produits sont définitives. Nous nous sommes heurtés à deux problèmes principaux lors de la manipulation de ces époxies: Tout d'abord, leur coefficient d'adhérence est très élevé : une fois cuites et solidifiées, elles adhèrent très solidement aux surfaces avec lesquelles elles sont en contact et il est quasiment impossible de les détacher sans risquer de briser les formes obtenues. Ensuite, ces deux produits sont plus liquides que les précédents avec une viscosité très faible. Ces deux propriétés nous ont empêchés de nous servir des deux types de moules précédemment utilisés pour les encapsulations : les moules métalliques car l'époxy collaient définitivement aux parois et les moules en plâtre car les mélanges pénétraient les parois poreuses. Cette situation a motivé la fabrication de nouveaux moules en téflon. Ce matériau très solide a la particularité d'être très peu adhérent. Des tests préliminaires ont

été menés et ont confirmé la viabilité d'une telle solution. Ces nouveaux moules ont été réalisés par l'atelier de génie mécanique de l'École Polytechnique.

2.5 Contrôleurs externes d'implants

Le système de stimulation précédent disposait de deux types de contrôleurs utilisés avec le neurostimulateur : les contrôleurs de test flexibles (CTF) et les contrôleurs portatifs simplifiés (CPS). Ces contrôleurs sont constitués en général d'un amplificateur de puissance inductive RF pour le transfert de l'énergie, d'un modulateur de signal RF pour la transmission des données, d'un encodeur de trame pour la conversion des paramètres en mots binaires et d'une interface utilisateur pour la sélection des paramètres ou des modes de fonctionnement. Les CTF, dont l'architecture est basée sur un ordinateur PC, sont utilisés communément pour des tests exhaustifs des implants. Ils permettent de passer en revue toute l'étendue des paramètres ainsi que les différents modes de fonctionnement de l'implant. Les CPS, comme leur nom l'indique, disposent d'une interface usager simplifiée et de fonctions limitées. Ils sont conçus pour être utilisés pendant les expériences en phases aiguë et chronique chez l'animal. Ils sont basés sur des microcontrôleurs, des petits écrans à cristaux liquides (LCD) et trois ou quatre touches de fonction [43].

Lors des expérimentations chroniques, les techniciens ont rencontré des difficultés avec le fonctionnement de ces contrôleurs externes. Ces derniers, soumis à des manipulations non adaptées à la fragilité relative du système lors de leur utilisation,

subissaient différents bris matériels au niveau des boutons poussoirs ou des ruptures de connections entre les différents blocs. Des interruptions de l'affichage étaient aussi observées. Ces défauts ont nécessité de multiples réparations des contrôleurs. Nous avons donc réalisé un contrôleur plus robuste et résistant. En effet, le principal changement apporté a été d'assembler le contrôleur en un seul bloc, contrairement à la version précédente en deux boîtiers différents où la partie de modulation et de transmission était séparée de la partie d'interface utilisateur et de contrôle. La communication entre ces deux blocs se faisait par des câbles externes dont les soudures se sont brisées à plusieurs reprises. Les deux modules ont été montés sur la même plaquette de montage réduisant tout risque de rupture de communication. Les boutons poussoirs ont aussi été remplacés et le bouton d'activation des stimulations a été changé pour un bouton pouvant rester enfoncé après son activation, permettant une manipulation plus aisée par l'usager.

L'alimentation du contrôleur provient d'un adaptateur DC 12V standard. Cette tension, après avoir été régulée, est utilisée pour la modulation et l'amplification. Elle doit aussi être transformée à 5V pour alimenter le PIC et l'écran LCD. Ce changement de niveau était précédemment effectué par une diode ZENER montée avec une résistance en série. La diode a été remplacée par un régulateur 5V (UA7805C) pour augmenter la fiabilité et éviter toute dissipation de courant dans la résistance. De plus, un dissipateur de chaleur a été placé auprès du régulateur, permettant ainsi d'éviter toute surchauffe du composant de régulation. La Figure 2-5 montre une photographie de ce nouveau contrôleur externe.

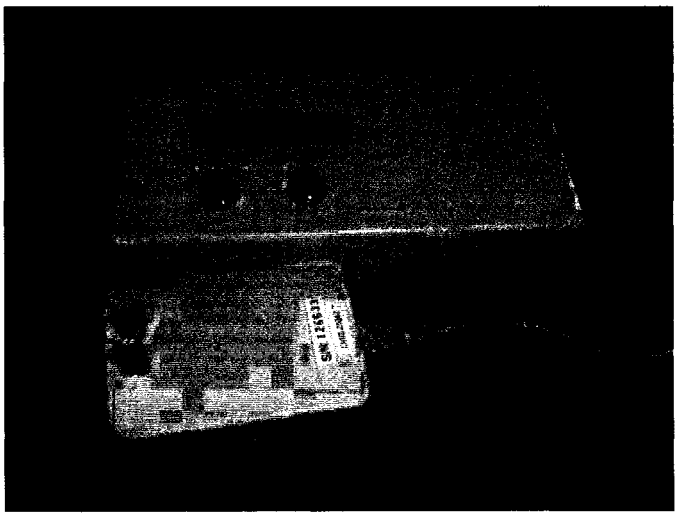


Figure 2-5 : Photographie du contrôleur externe.

Le démarrage de la stimulation sélective s'effectuait normalement, mais après qu'elle ait été arrêtée à partir du contrôleur externe, une tension relativement élevée (environ 2.6V) était toujours présente aux bornes de l'alimentation RF. Cette tension ne pouvait que provenir de la batterie qui était la seule alimentation restante dans le circuit. Les points de contact entre les deux alimentations ont été recherchés et après une étude isolée des amplificateurs de sortie, nous avons remarqué que l'application d'une tension sur la sortie d'un des amplificateurs dans un mode de stimulation entraîne une contre-réaction sur les entrées de l'autre amplificateur et ainsi l'induction d'une tension d'alimentation. Ceci nous a amené à changer l'amplificateur opérationnel (ampop) utilisé. Nous avons remplacé ce dernier par un autre ayant les mêmes caractéristiques, mais qui possède en plus une fonctionnalité de désactivation (*Shut Down)* permettant par l'intermédiaire d'une broche d'entrée dédiée de mettre les entrées et les sorties du composant en haute impédance.

L'opération de « *Shut Down* » de l'ampop pour la stimulation permanente est réalisée par le PIC par l'intermédiaire du signal SYS_OFF. Lors des stimulations sélectives ou en mode d'attente du systéme, le PIC applique un niveau logique bas sur cette broche permettant ainsi de désactiver les entrées/sorties de ce composant. Quant au circuit de stimulation sélective, un commutateur a été introduit afin de permettre de choisir l'origine de l'activation ou de la désactivation de l'ampop. Comme nous pouvons le voir sur la Figure 2-6, ce commutateur permet de choisir entre l'énergie RF ou un signal provenant du PIC (SHUT_DOWN_PIC).

Le choix de l'utilisation de l'énergie RF est justifié par le fait que celle-ci n'est présente que lors de la stimulation sélective, appliquant ainsi un niveau logique haut sur l'entrée du composant et l'activant, ou lors du transfert des paramètres.

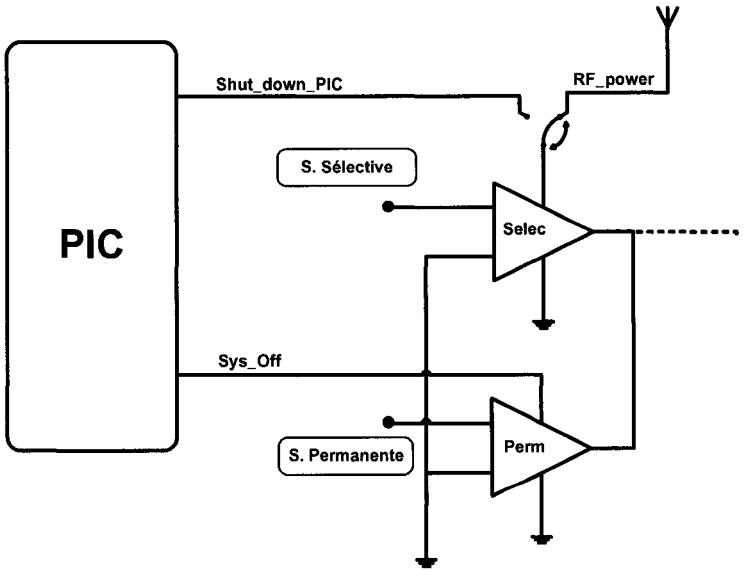
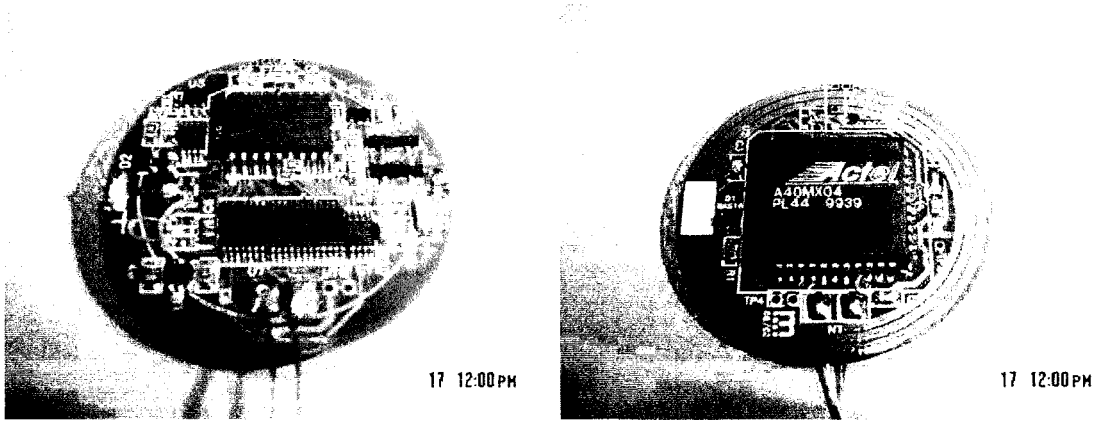


Figure 2-6 : Schéma de désactivation des amplificateurs par le PIC

Nous avons pu aussi remarquer lors de la validation en laboratoire une instabilité de la stimulation permanente. Celle-ci était causée par le fait que le PIC détermine son mode de fonctionnement en fonction des deux signaux DSR et SYS_ON. Lorsque le système est en mode de stimulation permanente, le contrôleur de bus est désactivé et ses entrées/sorties sont en mode haute impédance. Les signaux DSR et SYS_ON à l'entrée du PIC sont donc flottantes et toute variation inopportune de niveau sur un de ces signaux peut sortir le PIC de son mode de stimulation. Nous avons ainsi rajouté des résistances de mise à la masse (Pull-Down) sur ces entrées et forcé ainsi un niveau logique bas sur ces entrées lorsque le système est en mode de stimulation permanente.

Finalement, un nouveau circuit imprimé (PCB) a été réalisé pour la version proposée de l'implant. Les deux contrôleurs de stimulation utilisés précédemment ont été conservés : le FPGA 40MX04 d'Actel et le microcontrôleur PIC16F84 de Microchip. A l'exception des deux composants qui ont été remplacés, le contrôleur de bus initial par le SN74CBTD3348 et l'amplificateur de sortie par le TLV2450, la majorité des composants ont été empruntés à l'architecture précédente. Le schéma du circuit électronique de l'implant, les schémas du PCB ainsi que le nouveau programme assembleur du PIC sont disponibles à l'annexe A.

La Figure 2-7 présente une photographie des deux faces du microstimulateur assemblé à l'aide des composants montés en surface.



(a) (b)

Figure 2-7 : Photographie du nouveau microstimulateur (échelle 1/1) : (a) face supérieure, (b) face inférieure.

2.6 Conclusion

Nous avons tout au long de ce chapitre présenté les étapes qui ont mené à la réalisation du nouveau neurostimulateur sélectif et permanent. Nous avons identifié les modifications apportées par rapport à la version précédente du microstimulateur. Nous avons pu nous rendre compte des difficultés présentées par l'utilisation de multiples tensions d'alimentation dans un système électronique. Le prototype réalisé a fait l'objet d'expérimentations chroniques chez des animaux pour en valider le fonctionnement dans un environnement réel. Nous présenterons les résultats obtenus lors de ces expériences au chapitre 5 et nous analyserons les améliorations apportées par nos modifications au système précédent.

Nous verrons au cours des prochains chapitres les autres nouveautés apportées à tous les stimulateurs et leur implémentation dans un circuit intégré dédié en se servant de la technologie CMOS 0.18u.

CHAPITRE 3
TECHNIQUES DE STIMULATIONS SELECTIVE ET PERMANENTE ET NOUVEAU STIMULATEUR INTÉGRÉ

3.1 Introduction

La recherche dans le domaine de la réadaptation du système urinaire a engendré le développement d'un grand nombre de systèmes implantables dédiés à la récupération des fonctions vésicales. Cependant, la majorité de ces systèmes de stimulation électrique est caractérisée par certaines lacunes telles que :

- le manque de flexibilité des paramètres;
- les limitations au niveau du nombre de canaux et des types de stimulation;
- l'absence le plus souvent d'une interface usager conviviale;
- le manque de reprogrammabilité et de reconfigurabilité.

Le désir de remédier à ces manquements ainsi que les résultats pertinents obtenus avec les versions antérieures de notre système de stimulation nous ont motivé à concevoir et à réaliser une nouvelle version de stimulateur. Ce nouveau système intégré, répond aussi à un besoin croissant de miniaturisation et de minimisation de la consommation d'énergie des systèmes précédents.

Nous présentons dans ce chapitre un article qui a été soumis pour publication dans la revue « NEUROMODULATION». Il résume les différents travaux effectués lors de cette maîtrise et présente la nouvelle version intégrée de stimulateur.

L'article est organisé globalement de la même manière que ce mémoire de maîtrise: tout d'abord, dans la section 1, nous rappelons le fondement de la stimulation électrique neuronale. Nous présentons aussi les différentes techniques de stimulation ainsi que les systèmes déjà existants. Dans la section 2, nous décrivons l'architecture et présentons les différentes parties du systèmes de stimulations neurales sélective et continue. La section 3 est consacrée aux expériences de caractérisations in vivo chez l'animal du système de stimulation précédent. Nous y présentons le protocole expérimental utilisé ainsi que les résultats obtenus sur plusieurs chiens sur une période de plusieurs mois. La section 4 est entièrement dédiée au nouveau stimulateur intégré. Nous y présentons une description globale du système. Celle-ci est suivie par une présentation plus détaillée de l'architecture du microstimulateur. Nous décrivons ensuite le protocole de communication utilisé ainsi que la méthode de génération de stimuli utilisée. Finalement, nous validons le fonctionnement de la puce en présentant quelques résultats de simulation ainsi que divers résultats expérimentaux obtenus après la fabrication de la puce. Nous concluons ce travail en rappelant les bénéfices des stimulations sélective et permanente pour la réhabilitation du système urinaire, ainsi que l'étape franchie par la réalisation d'un nouveau système intégré dans la direction de la conception et de la réalisation du système de stimulation urinaire le plus complet et le plus flexible possible.

3.2 Dual stimulation techniques to recuperate the urinary bladder functions: chronic experiments in dogs and new implantable neural stimulator.

Abstract - Available electrical neurostimulation techniques for the recuperation of the bladder functions do not allow the adequate voiding due to the dyssynergia between the bladder and the sphincter. A new implantable stimulator, built with commercially available electronic components, is designed to overcome these difficulties. The proposed system, performs two types of stimulations: selective for bladder voiding and continuous to cancel the bladder hyperreflexia symptoms. Eight prototypes of the implant have been used in an experimental evaluation in chronic dogs to characterize the reliability and functionality of the new device. Also, a fully integrated extended version of the stimulator is achieved to monitor the electrodes-nerve contact which allows the detection of nerve impedance variations. The preliminary results of our chronic study show that the proposed stimulator provides significant improvement for bladder hyperreflexia curing and proved the efficiency of the selective stimulation by means of high frequency blockage in increasing the voided urine volume. In addition, the implemented full custom device provides reliable stimulation technique while its modular architecture makes the device an expandable multichannel stimulation system.

Index terms - Medical device, Electronic implant, Electrical stimulation, Selective and Continuous stimulation, Bladder controller, detrusor-sphincter dyssynergia, Detrusor hyperreflexia.

I. INTRODUCTION

Spinal-cord injured patients at the T12 level or higher can lose the control of their urinary bladder. These patients become unable to voluntarily evacuate the urine from filled bladders, and they often suffer from many complications such as detrusor hyperreflexia, which consists in a hyperactivity of the autonomic nervous system (ANS). The overfull bladder sends sensitive neural signals (SNS) to the spinal cord, where they travel upward until they are blocked by the lesion at the level of injury. Since these SNS cannot reach the brain, the reflex arc stays activated, increases activity of the sympathetic portion of the ANS and causes spasms. In most cases, incontinence resulting from dysfunctions of the bladder neck (urethral insufficiency) or by the detrusor instability (detrusor hyperreflexia), often occurs [1].

A. Electrical stimulation to recover the micturition function.

Several types of functional electrical stimulation (FES) have been introduced to allow the recovery of the voluntary control of the micturtion reflex at different sites of the urinary system. Four main stimulation sites have been investigated: the bladder muscle (detrusor), the pelvic nerves, the spinal cord and the sacral roots [2]-[7].

The stimulation technique of the detrusor did not allow to induce adequate voiding and the required high amplitude of the stimulation current may cause damages [8],[9].

FES have been applied to the pelvic nerves, but their relation with the pudendal nerves induced simultaneous excitation of the sphincter and the detrusor. This phenomena, known by Detrusor Sphincter Dyssynergia (DSD), induces high detrusor pressure that can eventually leads to incontinence or kidney failure. An alternative solution implying neurotomy of the pudendal nerve has been proposed but this surgical approach makes this technique less popular [7],[10].

The stimulation of the spinal cord has been performed, but using penetrating electrodes may cause the double activation of the bladder and the striated sphincter muscles [11],[12].

The last stimulation site, the sacral roots, is one of the most promising methods, but early conventional stimulation of sacral nerves induces once again the DSD [8]. Applying specific stimuli that allow to overcome the DSD attracted recently the researchers attention. In fact, the detrusor and the external sphincter muscles share the sacral nerves as a common innervation pathways. The autonomic afferent roots stimulate the sacral micturition reflex while the somatic efferent pathways (controlled by the brain) are responsible for the contraction of the external sphincter [1]. This stimulation method is the subject of the present work.

B. Sacral roots stimulation techniques

The intermittent stimulation of the sacral roots, also known as poststimulus voiding, has been described and largely used in patients [13]. By stimulating the sacral nerves in an adequate way, contractions can be induced in both the detrusor and the

sphincter, but only the striated sphincter muscle is able to relax between the stimuli, thus allowing voiding of the urine. The used stimulation pattern consists of intermittent pulse trains (typically, 3 to 6 seconds stimulation and 6 to 9 seconds stop). Good clinical results have been observed, but this intermittent stimulation method is characterized by a high intra-vesical pressure and high level of residue after stimulations that are damageable for the kidneys. Also neurectomy of nerves may be required in most cases [14].

Another sacral nerve stimulation technique is the sphincteric fatigue. In this case, the pudendal nerves are stimulated with high-frequency signals until induction of the sphincteric fatigue. It is followed by low frequency stimulation of the sacral nerves for voiding. This two-steps technique takes advantage of the difference between the contraction and the relaxation periods of the sphincter and the bladder. Furthermore, this concept, while allowing to avoid nerves section, produces comparable inefficient results to those obtained with patients following pudendal neurectomy [15],[16].

More than a decade ago, a selective blockage method has been introduced. It is based on the blockage threshold (or excitation) difference of the A-delta and A-alpha fibers. Multiple types of blockage techniques have been used : pudendal nerves collision [17], anodal block by the mean of the sacral roots stimulation [18],[19], and pudendal nerves high-frequency blockade [20],[21]. Recently, authors introduced the selective activation technique of the small nerves fibers in the sacral roots by combining a cathodic excitation of all fibers and a selective anodic blockage of the large fibers [22],[23]. This technique achieves hyperpolarization of the nerve membrane between the excitation application point and the external urethral sphincter to prevent the propagation of the

nerve action potential toward the sphincter. In fact, the large diameter fibers innervating the sphincter having a lower excitation threshold than those with small diameter, this blockage technique allows partial selective activation of the detrusor muscle thus inducing a little improved voiding.

More recently, selective detrusor muscle activation has been obtained by performing stimulation of the sacral roots with a signal composed of two distinctive trains of bipolar-current pulses. By applying this selective electrical stimulation, the somatic fibers driving the sphincter can be stimulated without causing simultaneous contraction of the bladder: a high-amplitude, low frequency train provokes detrusor muscle contraction while a low-amplitude, high-frequency train inhibits the external urethral sphincter contraction to allow micturition. This type of blockage consists of maintaining the nerves and their motor ends in a refractory state to impede them from contracting [24],[25]. This stimulation method allows bladder evacuation with a low-pressure voiding and low residual urine. It has been shown that the most efficient blockage frequency was around 600 Hz.

C. Available implantable stimulators

Few implantable electronic devices were introduced for clinical purposes to adress the bladder dysfunctions. The Avery® laboratories presented a stimultator dedicated to treatment of the retention [26]. Their first version produced a monophasic votage stimulation; a bipashic version was made available later. This is not used anymore. Finetech® presented in 1985 a triple transcutaneous stimulator generating monophasic

stimuli [27]. And in 1987, Physico-Med® introduced its version of the monocanal stimulator for the retention [28]. In 1997, Medtronic introduced devices to treat incontinence and several versions followed. Their InterStim® therapy is widely used worldwide for both purpose, incontinence and retention [29]. More recently, the VOCARE® bladder system introduced by NeuroControl, which is an updated version of Finetech, has been approved for the North American market [30]. This device requieres rhizotomies to achieve the bladder voiding.

In the past years, several implantable stimulators and their external controllers have been proposed by our research team [31]-[34]. Multiple stimulation techniques were used for restoration of voluntary bladder voiding. Recently, we introduced a new stimulation system, composed of an implantable device which is remotely powered and controlled by hand-held interface. Also a continuous stimulation is applied with a battery entirely dedicated to this mode to allow a longer lasting life of the implant. Prototypes, made with commercially available electronic components on a printed circuit board, have been realized and used to conduct chronic studies on dogs during eight months [35]. By the mean of a selective stimulation using the high frequency blockage, DSD is reduced or even avoided. In addition, the continuous stimulation demonstrated the feasibility to reduce the hyperreflexia.

These preliminary good results motivated us to improve our device and then to address the lack of features such as a wide range of programmable parameters, a user-friendly interface, and the waveform flexibility. In addition, we proceeded to build a fully integrated version of our stimulation system, to which we added new required features for

highly efficient stimulation system. In this paper, first we describe the implantable stimulator used in chronic tests in dogs. Brief description of the FES system will be given in section II. Section III includes the experimental protocol and the preliminary results obtained. In section IV, we present the new miniaturized system. Sections V and VI are dedicated to describe the experimental results and to conclude this work.

II. SELECTIVE AND CONTINUOUS NEUROSTIMULATOR

This stimulation system, build around off the shelf electronic components, is composed of an external controller, an electronic implant, and a bipolar cuff electrode wrapped around the sacral nerve.

A. The external controller

This part of the stimulator has three main features : 1) allows the user to choose between the two types of stimulation (continuous and selective), 2) selects the desired parameters (width, frequency, and amplitude), and 3) sends to the implant the needed power for its functionality. In the case of the selective stimulation, after the selection of the appropriate parameters, the stimulation pulses are generated continuously while the inductive link remains between the implant and the external controller. For the continuous stimulation, the external controller sends the parameters, then it is shut off to let the implant start generating permanently the low-frequency stimuli.

The external controller is a friendly user interface based on few push buttons, a LCD, and a finite state machine implemented in a commercially available microcontroller

(Figure 3-1). Once the parameters are chosen, corresponding Manchester encoded data are sent to the implant after being Amplitude Shift Keying modulated. Also, power amplifier is used to transcutaneously transfer the needed energy using inductive coupling technique to power up the implant.

B. The implant

The FES system allows two operation modes for the implant: selective stimuli generation and continuous stimuli generation. Each of these modes is handled by an independent stimulus generator (SG).

The block diagram of the implantable stimulator, shown in Figure 3-1, is composed of a data and power recovery block, two SGs, two bipolar current sources (BCS) and a switching stage to serve both SGs and to assure the bipolarity of the applied stimuli. The data recovery block allows the reconstitution of the data sent by the external controller and the recuperation of the energy to power up the selective stimulation block. The continuous stimuli generator (CSG) is entirely powered by an embedded battery with any interaction with the external controller. The selective stimuli generator (SSG), built in a FPGA (Field Programmable Gate Array), gets its energy from the external controller via the power recovery block. This FPGA contains also an internal controller which is in charge of data decoding and verification using a communication Cyclic Redundancy Check (CRC) technique. Furthermore, it generates the adequate commands to produce selective stimulation waveforms or transfers to the CSG the required parameters. This CSG is also built in a commercially available microcontroller (PIC16F84). After

reception and checksum verification of the parameters, the CSG generates commands to produce low amplitude stimuli for the continuous stimulation. Those commands are sent to the BCS for a digital to analog conversion and current stimuli delivery through an operational amplifier based current source.

The selective stimulation technique duplexes high and low frequency (HF & LF) stimuli alternately to respectively activate somatic A-alpha fibers which innervate the sphincter and parasympathetic A-delta fibers innervating the detrusor muscle. Figure 3-2a shows a typical waveform of this technique. The continuous stimulation is composed of a train of low-frequency pulses (Figure 3-2b). Also, biphasic stimuli are applied to avoid the accumulation of charges in tissues.

On the other hand, the selective stimulation necessitates high current amplitude (up to 2 mA in a 1Kohms impedance). In order to avoid the saturation of the current source when the nerve impedance increases, 5V DC voltage is regulated from the received 20 MHz carrier. The selective stimulation being powered up from the outside, this required high current level does not affect the lifetime of the battery. Also, special attention has been paid to isolate these two power supplies and to duplicate the area SGs to guaranty the right operation of the system. The dual SGs of the implant enroll two independent stimulation pathways with independent power supplies. We thus have an implant that can be either a high amplitude bi-frequency selective stimulator or a low power and low amplitude continuous stimulator. The implant is connected to the nerve via one bipolar cuff electrode.

C. The cuff electrodes

Usually the electrodes cuff is formed by two platinum foils inside a silastic trunk. This cuff is wrapped around the nerve and the platinum foils make electrical contacts between the nerve and the electrode. Two categories of electrodes were built by our research team [36]. While the first uses a shape memory alloy (SMA), the other one is based on a super elastic memory alloy facilitating surgery and optimizing contacts with nerves. SMA electrodes are easy to wrap and manipulate when kept at low temperature, but they automatically recover their original shape (cylindrical around the nerve) when heated at body temperature. Super elastic electrodes are naturally closed and strong enough to stay wrapped around the nerve if low stretching forces are applied. But over a given tension (apply by the surgeon) electrodes could be open and placed easily around the nerve.

III. CHRONIC EXPERIMENTS IN DOGS

The chronic study was conducted on 6 adult male mongrel dogs. Animals were subjected to laminectomy at the level of T10 vertebra and the spinal cord was sectioned under direct vision. The procedure was carried out under general anesthesia and aseptic techniques. At the same setting, a limited sacral laminectomy was performed and the sacral roots were identified. The extradural ventral sacral nerves supplying the urinary bladder and external sphincter were hooked and stimulated with an external pulse generator (SD9 Stimulator, Grass Medical Instruments). The intravesical pressure was

measured through a tri-way 7F catheter connected to a portable urodynamic analyzer (UDS-120, Laborie Medical Tech. Inc.) and a computer. After identification of the proper sacral root (S2 in dogs), a SMA electrode was wrapped around the nerve and the stimulator was implanted subcutaneously in the flank of the animal.

A. Experimental protocol in animal

Following the surgery, the dogs were kept on the following stimulation protocol:

- *No Stimulation Phase:* This phase includes the first two months of the study, where the dogs were kept on intermittent catheterization only without any stimulation.
- *Low-frequency only stimulation (standard stimulation):* During the third month, the animals were stimulated with low-frequency current pulses only. After the spinal-cord shock phase, the animals usually developed bladder hyperreflexic contractions with reduction of the bladder capacity. Cystometric evaluations of the intravesical and intraurethral pressure measurements with electromyographic (EMG) recording of the pelvic floor muscles were performed to determine the most suitable set of stimulation parameters for each dog. During the procedure, the bladder of each animal was filled with sterile saline solution until it leaks and then evacuated of half its capacity.
- *Selective and continuous stimulation:* Two weeks after the development of detrusor hyperreflexia, the continuous low frequency / low amplitude neuromodulation current was turned on to suppress detrusor hyperreflexia and the selective stimulation applied to induce micturition in these dogs. Different sets of selective stimulation parameters

for high-frequency blockage were applied in order to select the one that could give maximum bladder evacuation with high intravesical pressure and low intraurethral pressure. The dogs were stimulated with the programmed set of parameters twice a day. Expelled and residual urine volumes were measured. Also the applied parameters for the continuous low frequency / low amplitude neuromodulation current stimuli were selected on urodynamic basis; the parameters that gave the least effective change in the vesical and urethral pressures (Figure 3-3) with an associated EMG activity of the external urethral sphincter, were selected. Weekly cystometric study to monitor intravesical and intraurethral pressures and monthly intravenous urography (IVU) to visualize both kidneys, ureters and the bladder were performed for each animal. Voiding cystourethrogram (VCUG) with neurostimulation was carried out after IVU study.

- *Low-frequency only stimulation:* During the 7th month, daily stimulations were performed with low-frequency only current pulses to compare selective high-frequency blockage with low-frequency only stimulation. In the same time, the continuous low frequency/low amplitude neuromodulation current was stopped till the end of the study.
- *No stimulation phase:* This included the last month of the study where all forms of stimulation were stopped and the dogs were kept again on intermittent catheterization only.

B. Preliminary results from the chronic study

All the dogs completed the study. One dog spontaneously developed detrusor hyperreflexia (DH) 6 weeks after the surgery and before the application of neurostimulation. The other 5 dogs developed DH 1 to 4 weeks after the start of low-frequency stimulation only. With the application of continuous stimulation, DH disappeared from all the experimented animals within 2 to 4 weeks.

Before the development of the detrusor hyperreflexia (the period of the initial intermittent catheterization), the average functional bladder capacity was 256.5 ± 32.5 ml (Table 3). During the period of detrusor hyperreflexia, the average functional bladder capacity was significantly reduced to 127.1 ± 17.3ml. The average voided urine volume was 39.5 ± 5.3 ml representing 31.1 % of the mean total functional bladder capacity during that stage. After application of both continuous and selective stimulations, the average functional bladder capacity was significantly increased (120%) to 279.9 ± 39.4ml, compared to the functional bladder capacity during the stage of detrusor hyperreflexia. The mean voided urine volume was 247.1 ± 24.1 ml and this represents 88.3 % of the mean total functional bladder volume more than five time the voided urine volume during the hyperreflexia phase.

The 7^{th} month , the dogs were kept on low frequency only stimulation for one month and the continuous stimulation was turned off. During that period the mean functional bladder capacity was 262.9 ± 30.7 ml. The mean voided urine volume was 89.8 ± 13.1 ml and this represents 34.2 % of the mean total functional bladder capacity.

The mean residual urine volume was 173.1 ± 31.8 ml and this represents 65.8 % of the mean total functional capacity. Over the last month, when the dogs were kept on intermittent catheterization only, the mean bladder capacity was 253.4 ± 32.5 ml (Figure 3-4).

IV. NEW FULLY INTEGRATED STIMULATOR

The novel dual stimulation technique has been validated by off-the-shelf electronic components built in a double face printed circuit board and it does not allow the monitoring of the electrode-tissues contact. The obtained results motivated us to build a fully integrated microstimulator on chip together with the required measurement interface. The proposed chip has been implemented in CMOS 0.18 um technology.

A. System description

The fully integrated microstimulator delivers both stimulation types: selective and continuous. It also allows to generate flexible stimulation patterns and permits to characterize the impedance of the electrodes-nerve contact. The operation mode of the implant is selected by the external controller and the corresponding parameters are transmitted wirelessly. The selective stimulation mode and the measurement technique require much more energy than the continuous stimulation and are powered by the external controller. The continuous mode, as indicated by its name, implies a permanent power supply (embedded battery); thus two power supplies are used for this stimulator.

The waveform generated for each mode are completely reprogrammable by the means of parameters (amplitude, frequency, pulse widths, on and off times) stored in a Random Access Memory (RAM).

Selective stimulation may require biphasic and symmetric pulses of equal amplitudes and widths. However, asymmetric charge-balanced waveforms allow better control of electrochemical reactions at the electrodes and may suppress undesired physiological reactions such as increase of excitation thresholds [37] . In order to gain more stimulation flexibility, we propose a new method based on an arbitrary stimulation pattern to program and generate the stimulation waveforms (Figure 3-5). All stimulation parameters are selected by the external controller and stored in an internal RAM.

Dysfunctions of the implant reduce the safety and the reliability of the stimulation and lead to shorten the device lifetime. An efficient and safe way to monitor the different parts of the implant and to measure the impedance of the electrode-nerve contact is required. An impedance monitoring block has been integrated in the new microstimulator to detect any failure of the stimulation electrodes. The measurement principle necessitates to inject a current in the nerve and to measure the corresponding voltage developed between the two electrodes contacts. This voltage is first converted into a frequency by a Voltage Controlled Oscillator (VCO) and the resulting frequency is converted to digital serial values and finally sent to the external controller using Load Shift Keying (LSK) technique. This measurement block can be set in idle mode when not used to reduce the power consumption [38].

B. Architecture of the proposed integrated microstimulator

The integrated microstimulator is based on an expandable multichannel modular architecture (Figure 3-6). This microstimulator includes four main blocks.

- The internal controller retrieves the stimulation commands from the data and detects the operation mode. It regroups a Manchester decoder which is dedicated to recuperate the 300 KHz clock and the data frames from the received encoded signal. Also, a Data Recovery (DR) module with a Cyclic Redundancy Check (CRC) block is used to assure the integrity of received parameters. Finally, the internal controller identifies the stimulation channel and mode and transfers the parameters to the adequate Channel Stimuli Generator (CSG).
- The CSGs with embedded RAM generate digital values and control signals to the Stimulation Channels (SC). Once programmed, the SCs require only power to operate. The stimuli generation starts as soon as all the parameters are stored in the RAM but stops automatically if the external energy is suppressed. In this case, the continuous stimulationn starts using the embedded battery. To generate the arbitrary waveform, each sample value is selected from the RAM at a corresponding address and applied to the Stimuli Waveform Generator (SWG). The first seven bits of this command are used for the amplitude, while the last one indicates the polarity. So, a series of pulses with variable amplitude and polarity is delivered. Once the stimulation frequency is reached, the stimuli pattern is repeated.
- The SCs, fully analog, produce the needed stimulation current amplitude depending on the data received from the CSG. Each SC features 1) an 8-bit programmable

DAC and a wide swing, high output resistance voltage controlled current source. The DAC, controlled by the CSG block, is based on identical PMOS transistors mounted in series or parallel to deliver to the current mirror of the output stage the needed current value; 2) an output stage dedicated to generate the stimulation current and its direction via two pairs of transistors (NMOS and PMOS) used as switches; 3) an on-chip calibration circuitry providing efficient and reliable stimulation technique and allows to reduce mismatch errors related to the fabrication process, as well as those related to temperature variations.

- The impedance measurement module build around a VCO that converts the voltage obtained from the stimulated nerve into frequency and other building blocks such as a finite state machine (FSM), a counter and a parallel to serial converter. The VCO produces a signal frequency proportionnal to the monitored voltage, and number of cycles of this signal is counted during a period fixed by the FSM. The impedance measurement technique starts first by a setup time (T_setup) to calibrate the VCO. It is followed by an impedance evaluation period (T_eval) where low frequency current stimuli are applied by the SWG to the SC, then a voltage can be measured at the nerve contact. This voltage controls the VCO and the resulting signal is sent back to the internal controller where an impedance digital value is extracted. This digital value is then serialized and sent to the external controller.

C. Communication protocol

The communication between the external controller and the implantable microstimulator requires frames of different lengths with the same following model. First

a header (01111110) allows the DR module to detect the start of every new frame, then the code to select one or both stimulation channels to be activated is sent. Next, the command for the operation mode follows and the stimulation parameters are sent with the CRC code calculated on all the bits of the frame excepted the header and the channel.

D. Stimuli generation

After detection of the stimulation mode, the CSG stores the parameters in the specified RAM and transfers it to the SWG. Each SWG includes several building blocks such as frequency dividers, counters, comparators and a FSM. Table 3.4 summarizes the stimulation parameters. The high frequency blockage technique is based on a high frequency bipolar waveform of fixed pulse width (HFW) and amplitude (HFA). Each time the low frequency period (LFP) is attained, a pulse is produced with the adequate amplitude (LFA) and width (LFW). The Low and High Frequency Periods (LFP and HFP) are generated by counters. For the continuous stimulation, bipolar low frequency waveform is produced during the on time (Ton). A counter and a comparator are used for the frequency (Freq) and width of the pulse (PW). The stimuli amplitude is directly fed to the SG and during the off-time (Toff), the block is disabled.

E. Simulation and measurement results of the integrated microstimulator

Figure 3-7 shows the fabricated chip in a CMOS 0.18 um process, which occupies 4mm2. Figure 3-8 depicts the simulated results of the VCO during the evaluation of 4 different impedance values. In Figure 3-9, the output current and corresponding voltage of the digital to analog converter (DAC) used for the current source are presented. This

figure depicts all possible 8-bit digital values (D7-0). A maximum stimulation current value of 2.25mA is obtained. The DC current is applied to the output stage after being converted into a bipolar stimuli using the few switches. Figure 3-10 shows the reference current, the current at the output of the stimulation stage and the output voltage at a resistive load's terminals.

The experimental results of the fabricated chip prove its functionality as shown in Figure 3-10. The microstimulator generates a wide range of stimuli with variable parameters. The high frequency waveforms range from 294Hz to 75KHz while the low frequency ranges from 4.6Hz to 1.2KHz. The VCO frequency goes from 50Hz to 300KHz allowing the detection of short and open circuits. The pulse durations vary from 3us to 853us. Also, up to 32 8-bit values can be stored in the RAM, allowing to generate flexible stimulation waveforms.

V. CONCLUSION

The experimental validation on chronic dogs confirmed that combining both selective and continuous stimulations have great promises for bladder rehabilitation. Results of the stimulation strategies and dedicated implantable stimulators confirmed that the reliability of the selective high frequency blockage stimulation has a valuable solution for long term bladder voiding in paraplegics. Also, the neuromodulation technique, based on permanent low frequency and low amplitude current stimulation, offers a promising solution to cure the hyperreflexia with no drawback. On the other hand, the design and implementation of the integrated microstimulator, by combining multiple stimulation techniques with a proven efficiency, address at the same time the requirements that most

stimulation systems lack such as flexibility, reliability and reprogrammability. The new microstimulator and the reported experimental results in animals indicate that the proposed technique constitutes an adequate solution for a complete rehabilitation of the urinary bladder functions.

ACKNOWLEDGEMENTS

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC), the Kidney Foundation of Canada (KFOC), the Canadian Institutes of Health Research (CIHR) and the Canadian Microelectronics Corporation (CMC) for their support.

REFERENCES

[1] BUBACK, D., The use of neuromodulation for treatment of urinary incontinence, *AORN Journal*, 2001,73:176-190.

[2] WALTER, J., S., SIDAROUS, R., ROBINSON, C., J., WHEELER, J., S., WURSTER, R.,D., Comparaison of direct bladder and sacral nerve stimulation in spinal cats, *Journal of Rehab. Res. and Dev.,* 1992, 29, 2:13-22.

[3] HEINE,J.P., SCHMIDT, R.A., TANAGHO, E.A., Intraspinal sacral root stimulation for controlled micturition, *Invest. Urolo.,* 1977, 15:78.

[4] BRADLEY, W.E., TIMM, G.W. AND CHOU, S.N., A decade of experience with electronic stimulation of the micturition reflex, *Urol. Int.,* 1971, 26:283.

[5] HABIB, H.N., Experience and recent contributions in sacral nerve stimulation for both human and animal, *Br. J. Urolo.,* 1967, 39:73.

[6] SCHMIDT, R.A., BRUSCHINI, H., TANAGHO,E.A., Urinary bladder and sphincter responses to stimulation of dorsal and ventral sacral roots, *Invest. Urol,* 1979, 16:300.

[7] HALEEM, A., S., BOEHM, F., LEGATT, A., D., KANTROWITZ, A., STONE, B., MELMAN, A., Sacral root stimulation for controlled micturition : Prevention of detrusor-external sphincter dyssynergia by intraoperative identification and selective section of sacral nerve branches, *Journal of Urolo,* 1993, 149:1607-1612.

[8] TALLALA, A., BLOOM, J., W., QUANG, N., FES for Bladder : Direct or indirect Means?, *PACE*, 1987, 10:240-245.

[9] RIJKHOFF, N.J.M., WIKSTRA, H., VAN KERREBROECK, P.E.V. and DEBRUYNE, F.M.J., Urinary bladder control by electrical stimulation : review of electrical stimulation techniques in spinal cord injury, *Neurourol. Urodynam,* 1997, 16:39-53.

[10] HOLMQUIST, B., STAUBITZ, W.J., The role of the pudendal nerve in connections with electronic emptying of the neurogenic cord bladder in dogs, *Journal of Urol.,* 1964, 91:41-54.

[11] NASHOLD B.S., FRIEDMAN H., GLEN J.H., GRIMES J.H.,BARRY, W.F., AVERY R., Electromicturition in paraplegia, *Arch. Surg.,* 1972, 104:195.

[12] FRIEDMAN H., NASHOLD B.S. JR, GRIMES J., Electrical Stimulation of the Conus Medullaris in the paraplegic - A five year review, *In FT hambrecht, JB Reswick (eds): Functionnal electrical stimulation, New York and Basel: Marcel Dekker, Inc,* 1977, 173.

[13] BRINDLEY, G.S., POLKEY, C.S AND RUSHTON, D.N., Sacral anterior root stimulators for bladder control in paraplegia, *Paraplegia,* 1982, 20:365.

[14] BRINDLEY, G. S., An implant to empty the bladder or close the urethra, *Journal of Neurology, Neurosurgery, and Psychiatry*, 1977, 40:358-369.

[15] LI, J.S., HASSOUNA, M., SAWAN, M., DUVAL, F., ELHILALI, M.M., Long-term effect of sphincteric fatigue during bladder neurostimulation, *The journal of Urol.,* 1995, 153:238-242.

[16] SAWAN, M., HASSOUNA, M., LI, J-S., DUVAL, F., ELHILALI, M., Stimulator design and subsequent stimulation parameter optimization for controlling micturition and reducing urethral resistance, *IEEE Transactions on Rehabilitation Engineering,* 1996, 4, 1:39-46.

[17] SWEENEY J.D., MORTIMER, J.T. AND BODNER, D.R., Acute animal studies on electrically induced collision block of pudendal nerve motor activities, *Neurourolog. Urodyn.,* 1989, 8:521.

[18] BRINDLEY, G.S., AND CRAGGS, M.D., A technique for anodally blocking large nerve fibers through chronically implanted electrodes, *J. Neurol. Neurosurg. Psychiatry,* 1980, 43:1083.

[19] KOLDEWIJN, E.L., RIJKHOFF, N.J., VAN KERREBROECK, PH.E.V., DEBREYNE, F.M.J. AND WIJKSTRA H., Selective sacral root stimulation for bladder control: Acute experiments in a animal model, *Journa of Urol.,* 1992, 151:1674.

[20] ISHIGOOKA, M., HASHIMOTO, T., SASAGAWA, I. IZUMIYA, K. AND NAKADA, T., Modulation of the urethral pressure by high-frequency block stimulus in dogs, *Eur. Urol.,* 1994, 25:334.

[21] SHAKER, H. S., TU, L. M., ROBIN, S., ARABI, K., HASSOUNA, M., SAWAN, M., ELHILALI, M. M., Reduction of bladder outlet resistance by selective sacral root stimulation using high-frequency blockade in dogs : An acute study, *The Journal of Urol.,* 1998, Vol. 160: 901-907.

[22] ACCORNERO, N., BINI, G., LENZI, G.L., AND MANFREDI, M., Selective activation of peripheral nerve fiber groups of different diameter by triangular shaped stimulus pulses, *Journal Physiol.,* 1977, 273:539-560.

[23] FANG, Z-P. AND MORTIMER, J.T., Selective activation of small motor axons by quasitrapezoidal current pulses, *IEEE Trans. Biomed. Eng.,* 1991, 38:168-174.

[24] WOO, M.Y. AND CAMPBELL, B., Asynchronous firing and block of peripheral nerve conduction by 20Kc alternative current, *Bull. LA, Neuro. Soc.,* 1964, 29:87.

[25] SOLOMONOW, M., ELDRED, E., LYMAN, J. AND FOSTER, J., Control of muscle contractile force through indirect high-frequency stimulation, *Am. J. Phys. Med.,* 1983, 62:71.

[26] AVERY LABS, The micturition stimulator, *Avery Laboratories, Farmingdale, New York*, 1978.

[27] MAGASI, P., SIMON, Z., Electrical stimulation of the bladder and gravidity, *Urology International*, 1986, No. 41:241-245.

[28] PERKINS, T., A., Versatile Three-channel stimulation controller for restoration of bladder function in paraplegia, *Journal of Biomed. Eng.,* 1986, Vol. 8:268-271.

[29] MEDTRONIC. Interstim® therapy for urinary control. *Medtronic Neurological Inc., Minneapolis,* 1999.

[30] NEUROCONTROL CORPORATION VOCARE®, bladder system, implantable functional neuromuscular stimulator. *Neurocontrol Corporation*, Ohio, 1998.

[31] ARABI, K., SAWAN, M., Implantable multiprogrammable microstimulator dedicated to bladder control, *Med. and Bio. Eng. and Comput.*, 1996, 34:9-12.

[32] ROBIN, S., SAWAN, M., ABDEL-GAWAD, M., ABDEL-BAKY, T.M. and ELHILALI, M.M., Implantable stimulation system dedicated for neural selective stimulation, *Med. and Bio. Eng. and Comput.* 1998.

[33] SCHNEIDER, E., ABDEL-KARIM, A.M., SAWAN, M. and ELHILALI, M. New stimulation strategy to improve the bladder function in paraplegics: Chronic experimentents in dogs. *23rd IEEE Int. Conf. of Eng. in Medecine and Biology Society,Istanbul Turkey,* 2001.

[34] BOYER, S., SAWAN, M., ABDEL-GAWAD, M., ROBIN, S. et ELHILALI, M., Implantable selective stimulator improve bladder voiding : design and chronic experiments in dogs, *IEEE Transactions on Rehabilitation Engineering,* 2000, 8:764-470.

[35] BA, A., SCHNEIDER, E., ABDEL-KARIM, A.M., SAWAN, M., ELHILALI, M.M., Implantable dual stimulator to recuperate the bladder functions: Chronic experiments in dogs, *IFESS , Slovenia*, 2002.

[36] CRAMPON, M.A., SAWAN, M., BRAILOVSKI, V., TROCHU, F., New easy to install nerve cuff electrode using SMA armature, *Artificial Organs Journal,* 1999, 23, 5:392-395.

[37] STIEGLIZ T., MATAL T., STAEMMLER M., A modular multichannel stimulator for arbitrary shaped current pulses for experimental and clinical use in FES, *Proceedings of the 19th International Conference IEEE/EMBS, Illinois*, 1997.

[38] DONFACK, C., SAWAN, M., SAVARIA, Y., An implantable measurement technique dedicated to the monitoring of electrodes-nerve contact in bladder stimulators, *Medical & Biological Engineering & Computing,* 2000, 38:465-568.

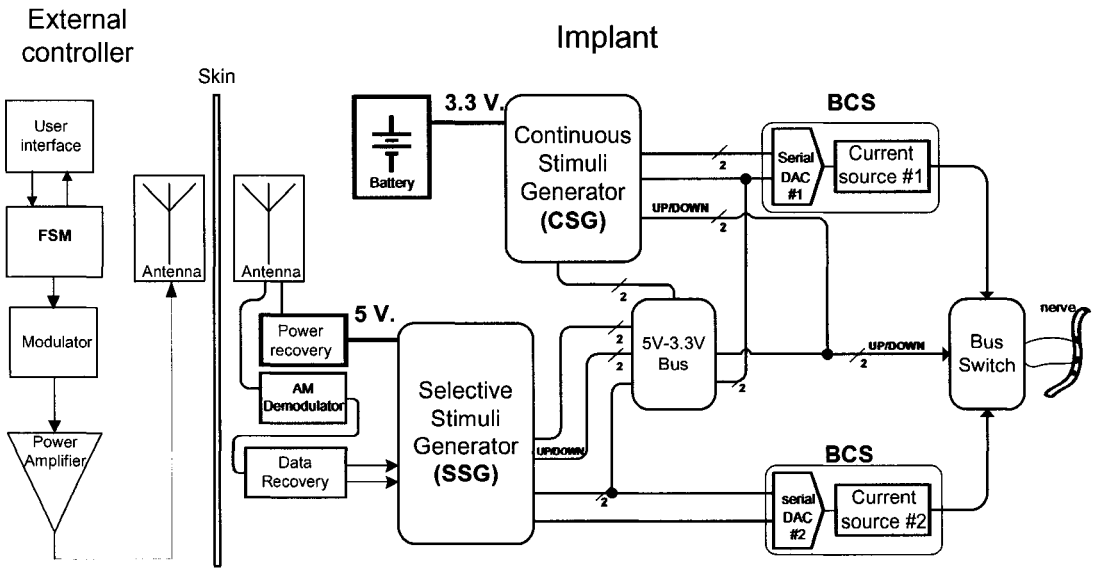


Figure 3-1: Block diagram of the stimulation system.

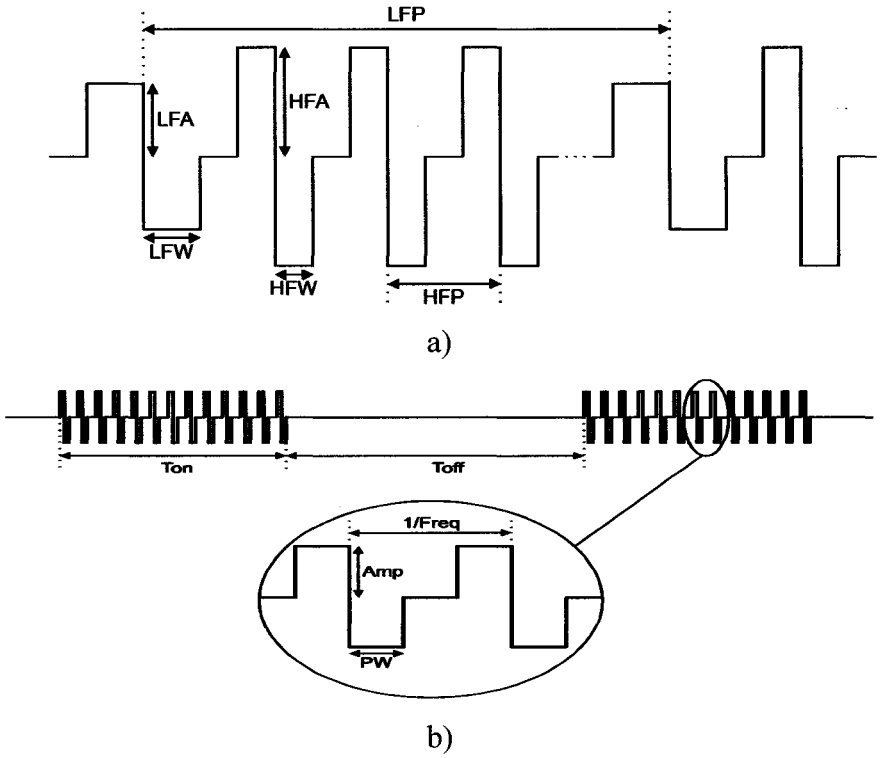
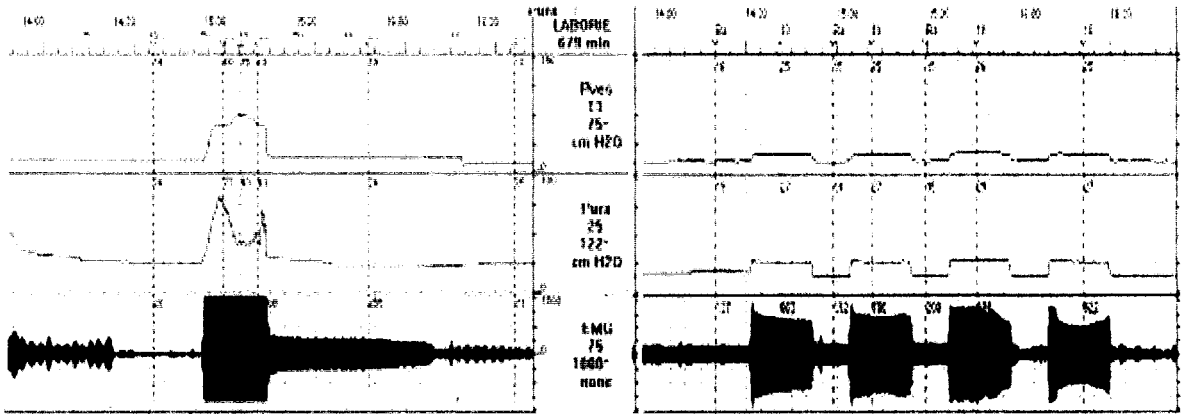


Figure 3-2: Stimulation waveforms: a) Selective stimulation ; b) Continuous stimulation. Parameters are described in Table 3.1 and Table 3.2 respectively.



a) b)

Figure 3-3: Typical CMG during: a) selective stimulation; b) permanent stimulation.

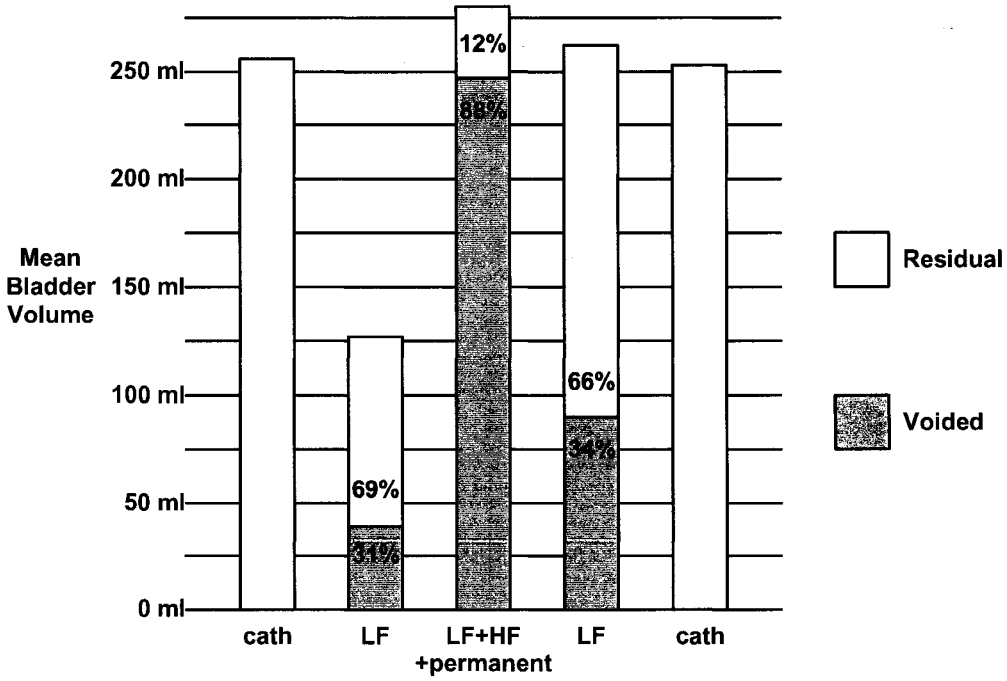


Figure 3-4: Average bladder volume and voided urine for three stimulation steps.

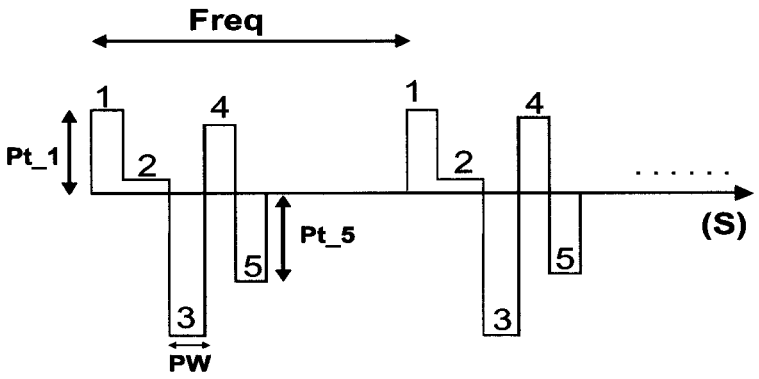


Figure 3-5: Flexible selective stimulation waveform. Parameters are described in Table 3.4.

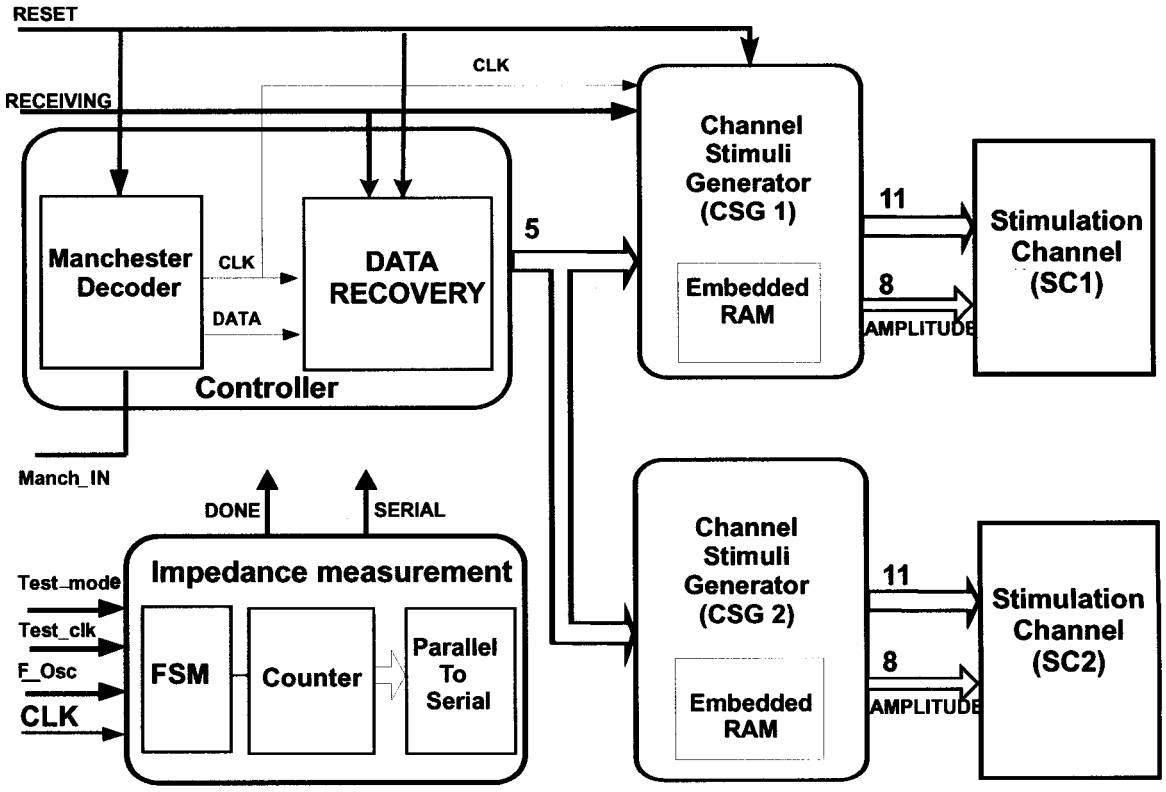


Figure 3-6: Block diagram of the integrated microstimulator.

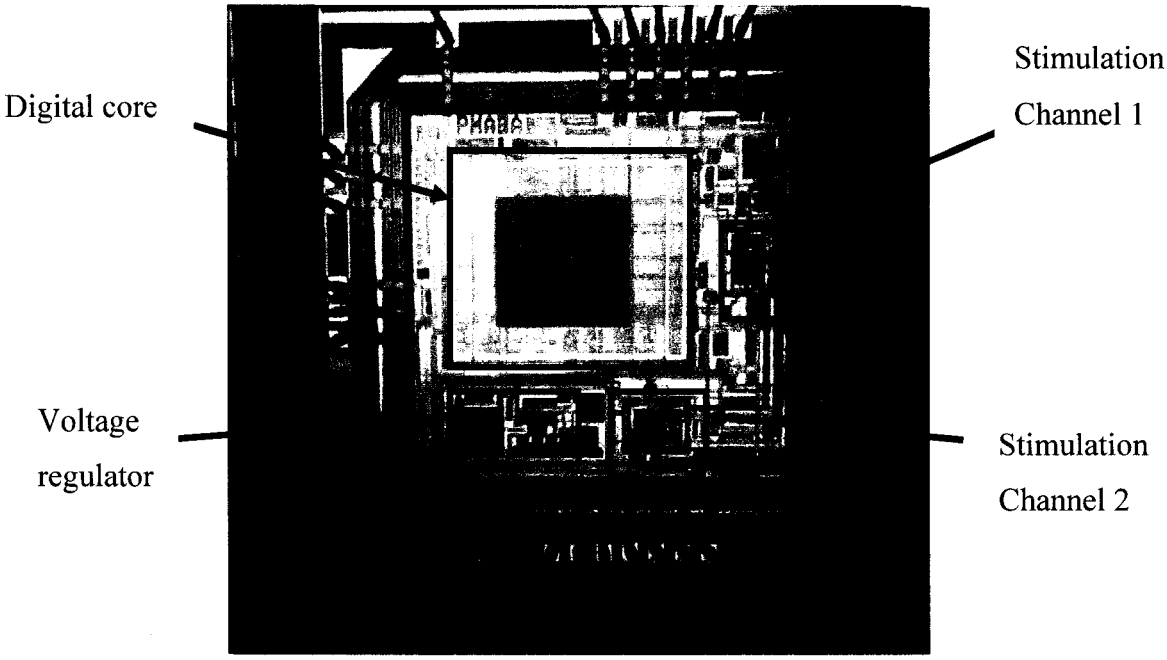
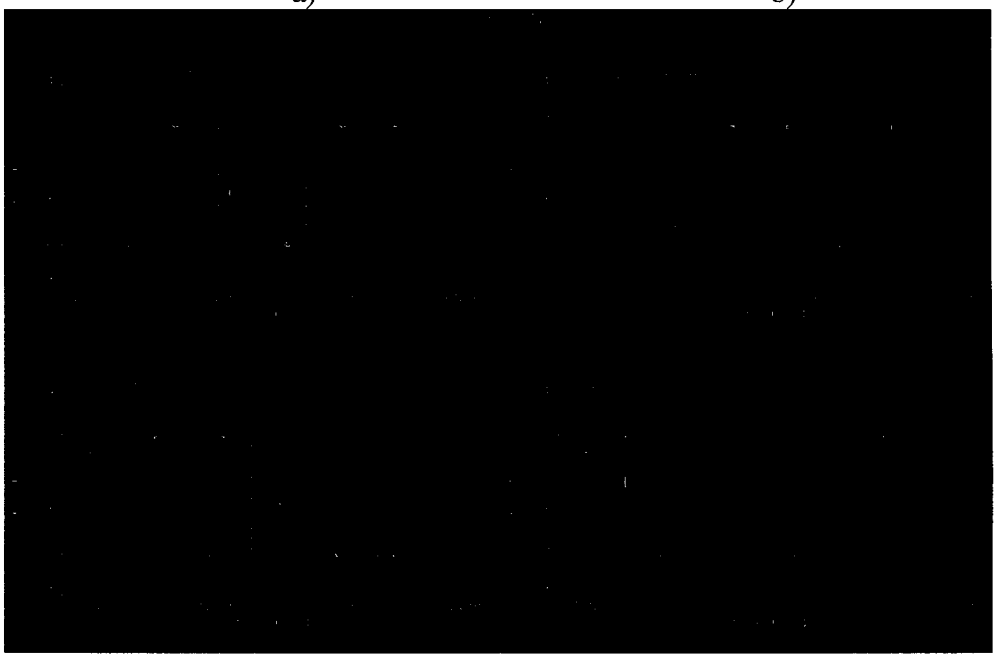


Figure 3-7: Chip microphotography. Important blocks are identified.

a) b)



c) d)

Figure 3-8: Simulation results of the impedance measurement's VCO a) Vin= 800mV; b) Vin= 850mV; c) Vin= 900mV; d) Vin=950mV.

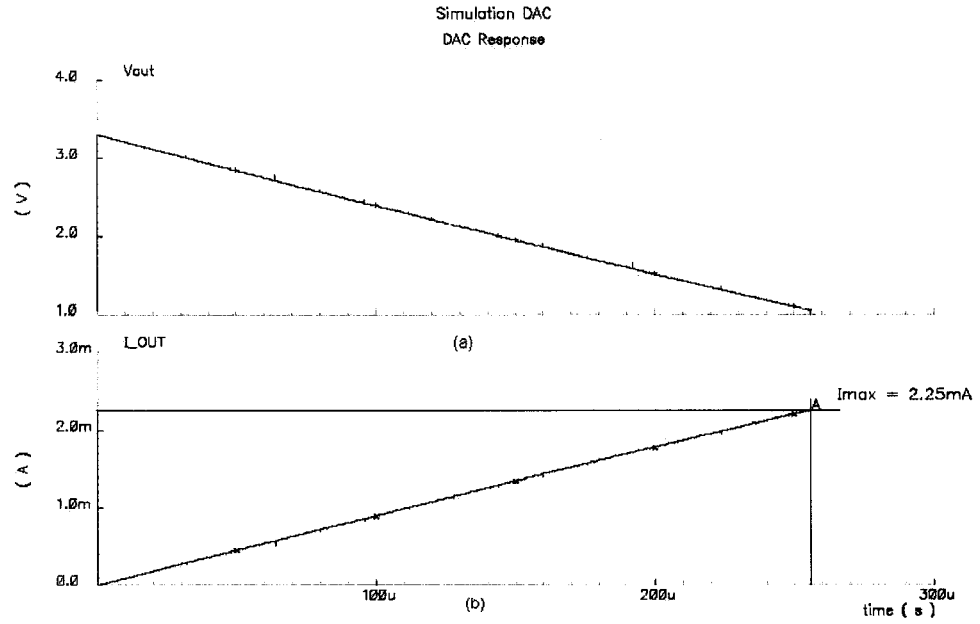


Figure 3-9: Simulation results of the DAC: a) Output Voltage; b) Output current after amplification.

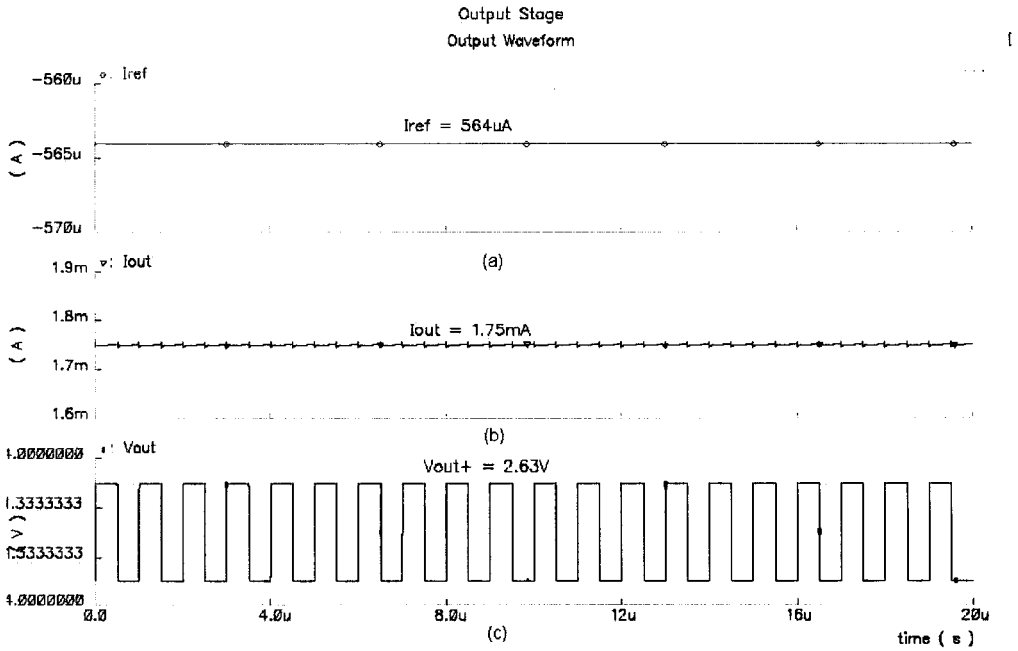


Figure 3-10: Simulation results of the output stage: a) Reference current from DAC; b) Output current after amplification; c)Waveform of the applied stimulation voltage.

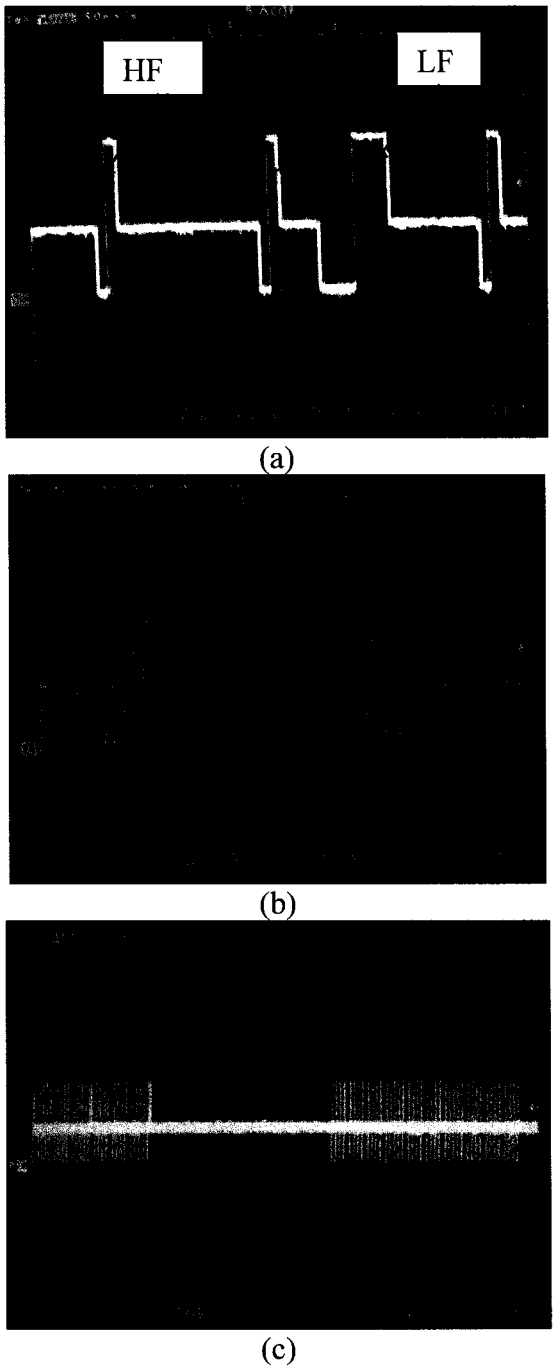


Figure 3-11: Measurement results from the integrated microstimulator: a) Selective high (HF) and low (LF) frequency stimulation, b) Stimulation with arbitrary pattern, c) Continuous stimulation with train of pulses.

Table 3.1:Selective stimulation parameters of implanted devices

	Low frequency			High frequency		
Parameter	**Amplitude**	**Frequency**	**Pulse width**	**Amplitude**	**Frequency**	**Pulse width**
Acronym	LFA	LFP	LFW	HFA	HFP	HFW
Units	mA	Hz	μs	mA	Hz	μs
Range	0 – 2	18 – 2000	3.3 - 210	0 - 2	290 - 2000	3.3 - 210
Typical value	0.9	30	175	1.3	600	100

Table 3.2: Continuous stimulation parameters of implanted devices

Parameter	**Amplitude**	**Frequency**	**Pulse width**	**On Time**	**Off Time**
Acronym	Amp	Freq	PW	Ton	Toff
Units	μA	Hz	μs	sec	sec
Range	0 - 1000	5 - 100	10 - 300	10 - 60	1 - 10
Typical value	200	30	175	20	10

Table 3.3: Average voided urine for three stimulation steps: chronic experiments in dogs

Protocol	**Average volume urine**		
	Voided (ml)	**Residual (ml)**	**Voided %**
Catheterization	N/A*	256.5 ± 32.5	N/A*
Low Frequency (3rd Month)	39.5 ± 5.3	87.6 ± 13.6	31.1
Selective and Continuous	247.1± 24.1	32.8 ± 17.6	88.3
Low Frequency (7th Month)	89.8 ± 13.1	173.1 ± 31.8	34.2
Catheterization	N/A*	253.4 ± 32.5	N/A*

*** N/A= Not attempted**

Table 3.4: Stimulation parameters of the integrated microstimulator

Type	**Command**	**Parameters (bits)**					
Continuous	10101010	LFP 8	LFW 8	AMP 8	Ton 4		Toff 4
Impedance measure	11001100	Freq 8	PW 8	AMP 8	Teval 4		Tsetup 4
Selective	00001110	LFP 8	HFP 8	LFW 8	HFW 8	LFA 8	HFA 8
Flexible	01110000	Nb_pt 4	Freq 8	PW 8	Pt_1 8	Pt_2 8	Pt_x 8

3.3 Conclusion

Dans ce chapitre, nous avons présenté l'article qui a été soumis pour publication dans la revue « NEUROMODULATION». Nous y avons résumé les différents travaux effectués lors de cette maîtrise et présenté la nouvelle version intégrée de stimulateur. Nous présentons dans le chapitre suivant les détails de l'architecture du stimulateur intégré ainsi que les étapes qui ont conduit à sa réalisation.

Nous présentons ensuite les différents résultats obtenus suite aux tests en laboratoire des deux versions de microstimulateur.

CHAPITRE 4
NEUROSTIMULATEUR PROGRAMMABLE INTÉGRÉ

Nous apportons dans cette section un complément à l'article présenté dans le chapitre précédent. Nous présentons en détail la conception et la réalisation du neurostimulateur programmable intégré.

4.1 Spécifications

Les deux fonctions de stimulation déjà présentes dans les versions antérieures seront retrouvées dans le nouveau système intégré. Ainsi le stimulateur sera en mesure de générer des signaux de stimulation sélective et permanente avec les mêmes paramètres décrits dans les chapitres précédents. Nous introduisons aussi, dans un souci de flexibilité des types de stimulation, une architecture permettant d'effectuer une nouvelle catégorie de stimulations. Cette stimulation, dite flexible, permettra à l'usager de générer, par l'intermédiaire de points sauvegardés dans une mémoire à accès aléatoire (RAM), une vaste étendue de formes d'ondes. Ces formes d'ondes pourront ne pas être obligatoirement symétriques ou bipolaires. Nous pourrons ainsi évaluer les réactions des nerfs stimulés, lorsqu'ils sont soumis à des charges qui peuvent induire de meilleures réponses sur le plan physiologique.

La Figure 4-1 présente un exemple de formes d'ondes qui pourra être générée dans ce mode de stimulation.

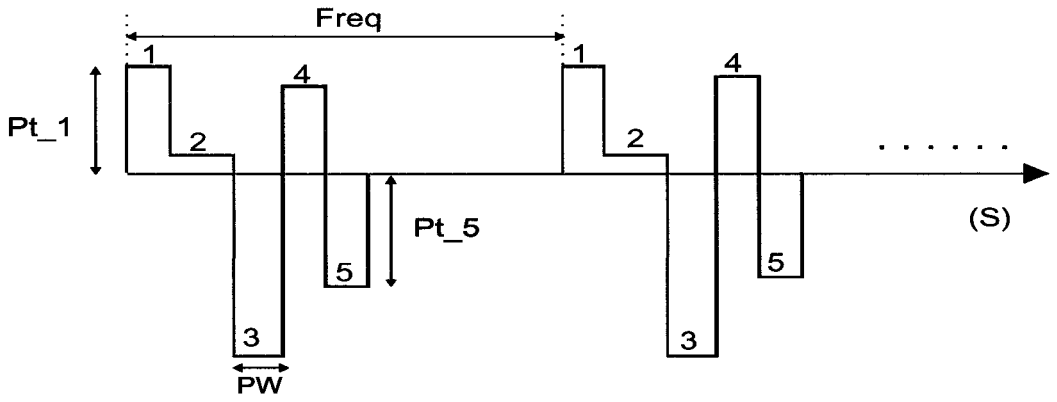


Figure 4-1: Forme d'onde flexible générée à l'aide de la nouvelle architecture du stimulateur

Les paramètres de ce nouveau type de stimuli sont décrits dans le Tableau 4.1. Nous pouvons remarquer que nous avons introduit un paramètre *Sign* qui comme son nom l'indique représente la polarité de l'amplitude du point considéré.

Tableau 4.1: Paramètres du nouveau type de stimulus

Paramètre	Fréquence (1/période)	Largeur d'impulsion	Amplitude	Signe
Acronyme	Freq	PW	Amp	Sign
Unité	Hz	µs	mA	N A*
Plage	290 – 2000	3.3 - 210	0 – 2	0-1
Résolution	1	10	0.01	N A*

*N A : Non Applicable

En plus de la génération de stimuli flexibles, nous avons ajouté un bloc de mesure d'impédance. Cette technique, mise au point précédemment par notre équipe de recherche [16], permet de mesurer toute variation d'impédance aux « bornes » du nerf stimulé. Elle permettra non seulement de vérifier l'état des tissus lorsque soumis à la stimulation électrique, mais aussi de détecter tout bris des électrodes de stimulation. Le

système devra être capable de mesurer une plage de valeurs d'impédance allant du court-circuit au circuit ouvert et de renvoyer la valeur obtenue à l'usager.

L'implant recevra ses paramètres de configuration et ses commandes de stimulation de l'extérieur. Nous utiliserons un contrôleur externe du type CTF avec une interface PC permettant de caractériser toutes les fonctionnalités du stimulateur. Les données reçues sont codées en format Manchester et regroupent l'horloge ainsi que les commandes. Nous utiliserons la même fréquence de fonctionnement (300KHz) que dans les versions antérieures.

La fonction de récupération d'énergie sera implantée à l'extérieur de la puce. Nous prévoyons cependant une technique de double alimentation afin de pouvoir reproduire les conditions d'utilisations normales, notamment lors de la stimulation permanente.

Cette version comprendra deux canaux de stimulation totalement indépendants. Le système devra être cependant conçu de façon modulaire afin de pouvoir y greffer facilement de nouvelles fonctionnalités ou d'augmenter le nombre de canaux. Le design combinera des modules numériques et analogiques et sera réalisé et fabriqué dans la technologie CMOS 0.18um.

4.2 Design du circuit intégré

4.2.1 Modes de fonctionnement

Le système reçoit ses commandes et ses paramètres d'un contrôleur externe. Après le décodage des données et la reconstitution de l'horloge par le décodeur Manchester intégré, l'implant identifie le mode de stimulation ainsi que les canaux de

stimulation à activer (une vérification préliminaire est effectuée pour s'assurer que les canaux ne sont pas déjà actifs). Les deux canaux peuvent être activés en même temps et effectuer différents types de stimulations ou des stimulations identiques avec des paramètres différents. Cependant une stimulation permanente ne peut avoir lieu en même temps que n'importe qu'elle autre type de stimulation car les alimentations sont différentes : la stimulation permanente possède une alimentation dédiée et elle attend que l'alimentation principale soit éteinte avant de démarrer.

Globalement, l'implant peut être décomposé en deux blocs. Un contrôleur interne intégré qui décode, transfert les paramètres de stimulation et active les canaux de stimulation. Le deuxième bloc est constitué des canaux de stimulation indépendants qui génèrent les formes d'ondes de stimulation souhaitées.

Chaque canal peut générer les différents types de stimulation décrits précédemment et permet de mesurer l'impédance du contact electrode-tissus :

- Stimulation sélective : un signal bipolaire constitué de deux ondes superposées, avec chacune leur amplitudes et leur fréquences spécifiques, est appliqué au nerf pour agir sélectivement sur les deux organes du système urinaire (sphincter et détrusor).
- Stimulation permanente : un signal périodique de stimulation permanente, à basse fréquence et à faible amplitude, est employé. Il est caractérisé par sa bipolarité et ses temps d'activation et d'inactivation programmables.
- Stimulation sélective flexible : le canal produit une forme d'onde en alignant successivement des points d'amplitude et de signes variables et utilisant une durée

d'impulsion fixée. Chaque fois que la fréquence de fonctionnement définie par l'usager est atteinte, la forme d'onde de stimulation est reproduite.

- Mesure d'impédance : le canal envoie tout d'abord un signal de calibration au nerf pour initialiser le système. Ce signal est suivi d'un courant à basse amplitude qui va permettre de générer une tension aux bornes du nerf. De cette tension sera ensuite déduite la valeur de l'impédance qui sera finalement transmise à l'usager de façon sérielle.

4.2.2 Protocole de communication

Un protocole de communication entre le contrôleur externe et l'implant adapté aux besoins de flexibilité requis pour cette architecture a été mis en place. Ce protocole utilise la technique d'encapsulage de données et est composé de quatre types de « trames| » de longueurs différentes mais qui présentent une structure globale commune. Elles sont constituées d'une entête (huit bits), des canaux à activer (deux bits), de la commande (huit bits), des paramètres (nombre de bits variables) et pour terminer, d'une signature (CRC: Cyclic Redundancy Check) pour la détection d'erreurs [67]. L'entête "01111110" identifie le début de la trame tandis que le CRC permet de détecter toute erreur de transmission dans le mot de commande et le mot de données. Le principe utilisé est celui de la division polynomiale avec la fonction "X^8+1" comme polynôme réducteur [67]. Le mot de commande permet de préciser le mode de fonctionnement du système (Sélectif, permanent, mesure d'impédance ou sélectif flexible). Le mot de données contient les paramètres spécifiques à chaque type de stimulation d'où la

différence de longueurs des trames. Afin d'éviter que l'entête ne se retrouve dans les données, il est nécessaire de rajouter dans la trame le bit de remplissage '0' après chaque séquence de cinq '1' consécutifs au niveau de l'émetteur. Ces '0' de bourrage seront ignorés par le récepteur qui reconstituera exactement la trame initiale de stimulation. Cette technique est inspirée du protocole standard de communication X25 [67].

La Figure 4-2 présente les trames de stimulation utilisées pour communiquer avec l'implant tandis que le Tableau 4.2 décrit les paramètres constituants ces trames.

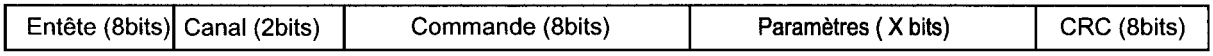


Figure 4-2: Trames de stimulation pour le nouveau stimulateur intégré (Tableau 4.2 fournit les valeurs des paramètres)

Tableau 4.2: Description des paramètres de stimulation

Type de stimulation	Paramètres					
Permanente "10101010"	LFP (8 bits)	LFW (8 bits)	AMP (8 bits)	Ton (4 bits)	Toff (4 bits)	
Mesure Z "11001100"	Freq (8 bits)	PW (8 bits)	AMP (8 bits)	Teval (8 bits)	Tsetup (4 bits)	
Selective "01110000"	LFP (8 bits)	HFP (8 bits)	LFW (8 bits)	HFW (8 bits)	LFA (8 bits)	HFA(8 bits)
Selective flexible "00001110"	Nb_pt (4bits)	Freq (8 bits)	PW (8 bits)	Pt_1 (8 bits)	Pt_2 (8 bits)	Pt_x (8 bits)

4.2.3 Architecture du système

La Figure 4-3 présente le diagramme bloc de l'implant. Il peut être décomposé en quatre blocs principaux.

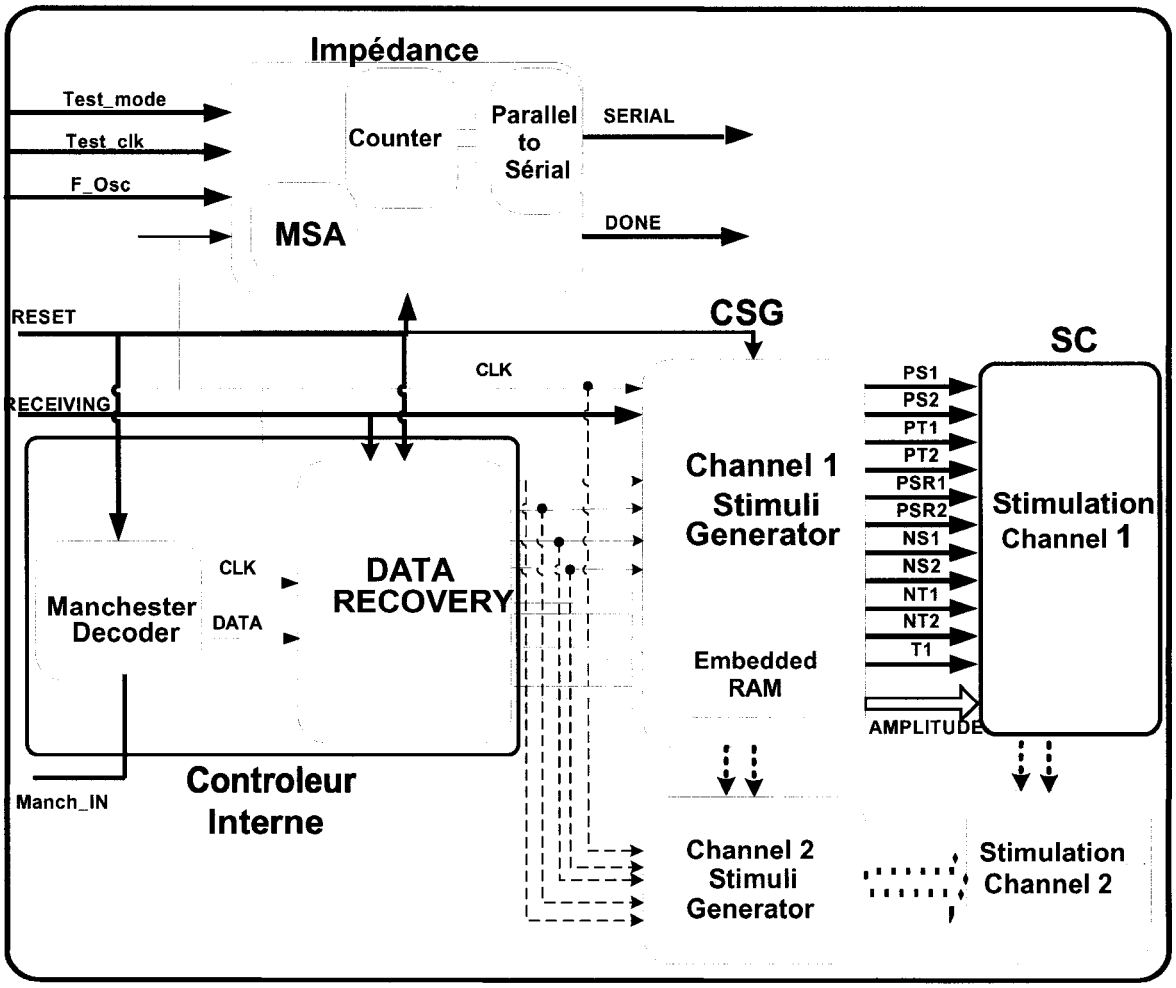


Figure 4-3: Diagramme bloc du système global

a) Contrôleur interne

Le contrôleur interne est le bloc d'entrée du système. Ce bloc est responsable de la récupération et du décodage des données de stimulation. Il est aussi en charge de l'identification des modes et des canaux de stimulation. Il est composé d'un décodeur Manchester et d'un bloc de récupération de données *(Data Recovery).*

Le décodeur Manchester décode le signal binaire reçu en format Manchester du contrôleur externe. Une configuration à trois bascules D et un circuit de délai de ¾ de période sont utilisés [54]. Le décodage de ce signal Manchester génère l'horloge globale du système à 300KHz et permet de récupérer de façon sérielle la trame de données.

Le module *DATA RECOVERY* quant à lui permet d'identifier le mode de stimulation ainsi que les canaux à activer. Il est aussi en charge du transfert des paramètres de stimulation vers ces canaux. La Figure 4-4 présente le diagramme bloc de ce module.

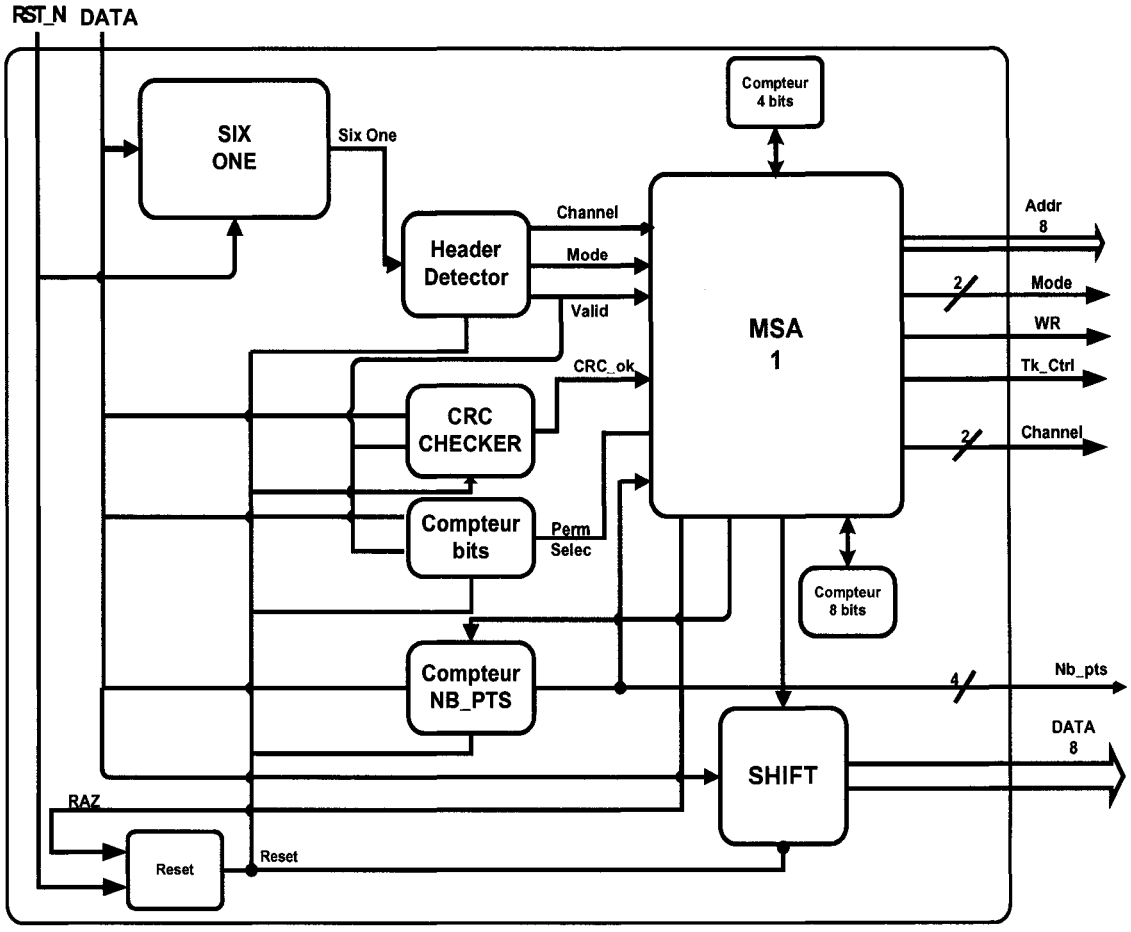


Figure 4-4: Diagramme bloc du module de récupération de données

En entrée du bloc, le module SIX ONE permet de détecter les six '1' contenus dans l'entéte de la trame. Par le signal *Six One* il active le module HEADER DETECTOR qui identifie les canaux de stimulation et decode la commande de fonctionnement. Lorsqu'une combinaison valide (Canal + Mode) est trouvée, le bloc génére un signal *Valid* qui va activer le début du transfert des paramètres de stimulation au canal spécifié. Après génération de ce signal, le module HEADER DETECTOR reste dans un état d'attente jusqu'à ce qu'il y ait une réinitialisation du système.

Une des particularités du protocole de communication est d'introduire des zéros de bourrage dans la trame pour éviter de retrouver six '1' consécutifs dans celle-ci. Nous devons ainsi en tenir compte dans la récupération des données. Le bloc SIX ONE produit aussi un signal qui indique la détection de cinq '1' consécutifs. La combinaison de ce signal et du signal *Valid* décrit plus haut permet d'obtenir un nouveau signal qui permet de désactiver le système pendant un cycle d'horloge suite à la détection de cinq '1' consécutifs, mais cela uniquement après la détection d'une entête valide.

Le signal V*alid* permet aussi de déclencher la vérification du CRC de la trame ainsi que le début du compte des bits de la trame. Le bloc CRC CHECKER va, au fur et à mesure que la trame est décodée, effectuer une division polynomiale de la trame et à la fin, comparer le reste (mot de huit bits) avec la valeur « 00000000 ». Si le reste est égale à zéro, le CRC est valide et un signal *Crc_ok* est activé. Le compteur de bits permet de détecter les fins des trames de stimulation sélective, permanente et mesure d'impédance. Le bloc COMPTEUR NB_PTS est utilisé exclusivement dans le mode de stimulation sélective flexible pour déterminer le nombre de points utilisés pour la génération du

« pattern » de stimulation. La sortie de ce bloc est un signal codé sur quatre bits, *Nb_pts*, qui est relié directement aux canaux de stimulation.

La coordination entre tous ces blocs est gérée par une machine à états, MSA1. Après réception du signal valide du module HEADER DETECTOR, MSA1 active le chargement des bits de la trame dans le registre à décalage (SHIFT). SHIFT est un registre de 9 bits, dont les huit bits de poids les plus significatifs sont utilisés pour transférer les paramètres de stimulation aux canaux. En même temps que le chargement du registre, la MSA1 permet de compter le nombre de bits stockés dans ce registre.

La MSA1 a aussi le contrôle des RAM des canaux et par l'intermédiaire des signaux *WR* et *Addr* écrit les paramètres de stimulation par bloc de huit bits dans les mémoires des canaux concernés. L'accès à la RAM des canaux est rendu possible par l'intermédiaire du signal *Canal* qui est codé sur deux bits. Avant d'activer un canal, MSA1 s'assure par que le canal désigné n'est pas déjà actif. Dans ce cas, MSA1 réinitialise le système et attend le prochain bloc de paramètres valides. Lorsque l'écriture des paramètres en RAM est achevée, la MSA1 envoie un signal de « prise en charge » au canal actif. Ce signal, *Tk_ctrl* indique au canal qu'il a le contrôle pour la génération des stimuli des différents mode de stimulation. Le cycle d'horloge suivant la prise de contrôle par le canal, la MSA1 produit un signal de réinitialisation *(RAZ)* du système. Ce système est combiné au signal d'entrée *Reset* pour activer tous les blocs du module DATA RECOVERY. Cette combinaison permet un double contrôle de la réinitialisation du bloc. À la fin de cette séquence, la MSA1 se met dans un état d'attente de nouveaux paramètres valides. La Figure 4-5 présente le diagramme d'états simplifié de la MSA1.

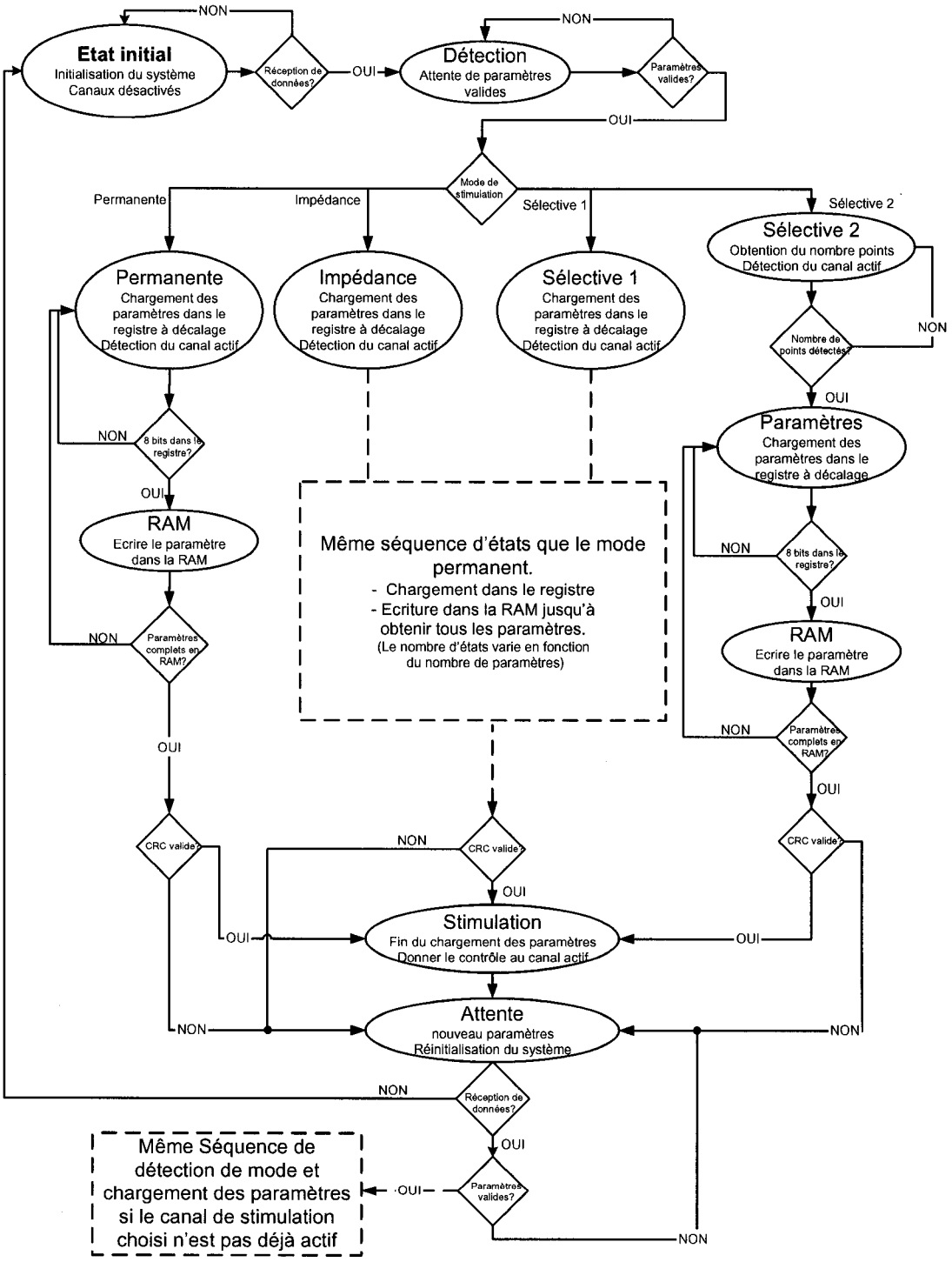


Figure 4-5: Diagramme d'états de la machine à états contrôlant la récupération de données

On peut remarquer sur le diagramme une séquence qui est indiquée en pointillés. Celle-ci est en fait la réplique de la séquence de chargement des paramètres et de l'écriture de ceux-ci dans la RAM. Nous n'avons indiqué la séquence complète que dans le cas de la stimulation permanente. Dans le cas de la stimulation sélective flexible, cette séquence diffère car elle inclut un état préliminaire d'obtention du nombre de points de stimulation.

b) Générateur de stimuli (Channel Stimuli Generator)

Après la récupération et le transfert des paramètres de stimulation par le contrôleur interne, le générateur de stimuli pour chaque canal est en charge de produire les signaux qui contrôlent la génération des stimuli souhaités. La Figure 4-6 montre la structure de ce bloc.

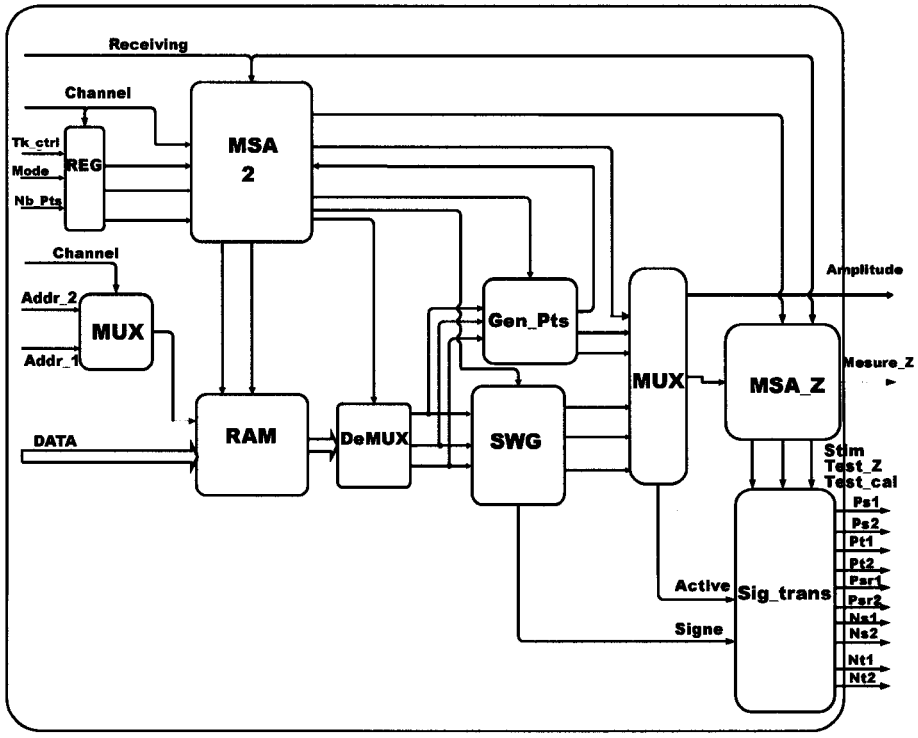


Figure 4-6: Diagramme bloc du générateur de stimuli d'un canal

Nous avons doté notre système de deux canaux de stimulation. Nous avons ainsi deux générateurs de stimuli identiques, mais indépendants. Ils communiquent par l'intermédiaire des mêmes bus de données avec le contrôleur interne et sont activés sélectivement par le signal *canal* dont les deux bits ne commandent chacun qu'un bloc de façon exclusive.

Le registre de données en entrée (REG) est utilisé pour conserver le mode de stimulation, le signal *Tk_Ctrl* provenant du contrôleur interne, et le cas échéant le nombre de points pour la stimulation sélective flexible. REG est nécessaire au bon fonctionnement du bloc, car après l'envoi des données décrites plus haut, le contrôleur interne traite de nouvelles trames, et des variations de niveau peuvent survenir sur ces signaux. L'activation du REG est contrôlée par le signal *Channel* activant le générateur de stimuli. Un multiplexeur activé par le même signal est aussi utilisé en entrée pour permettre, une fois l'écriture des paramètres de stimulation dans la RAM terminée, de donner le contrôle de celle-ci à la machine de contrôle du bloc MSA2.

La RAM utilisée est un bloc de mémoire avec deux ports de huit bits d'écriture et de lecture). Elle est divisée en trente-deux adresses de huit bits de données permettant ainsi de sauvegarder tous les paramètres de stimulation et jusqu'à trente points de stimulation flexible. Ses opérations en écriture sont commandées par le contrôleur interne tandis que celles en lecture dépendent exclusivement de MSA2. Le contrôle global du fonctionnement du générateur de stimuli est réalisé par MSA2. Elle orchestre la lecture des paramètres de stimulation à partir de la RAM. La Figure 4-7 décrit le diagramme d'états simplifié de MSA2. La MSA2 sort de son mode initial sur réception du signal

d'activation du canal (*Channel*). Dès la détection du signal *Tk_ctrl*, elle commence la lecture des paramètres à l'aide des signaux *Rd_n et Addr* qui commandent respectivement la lecture et les adresses de la mémoire. Les sorties de celle-ci sont envoyées à un démultiplexeur qui sert de registre et qui permet aussi d'envoyer les paramètres de stimulation adéquats aux blocs générateurs de formes d'ondes.

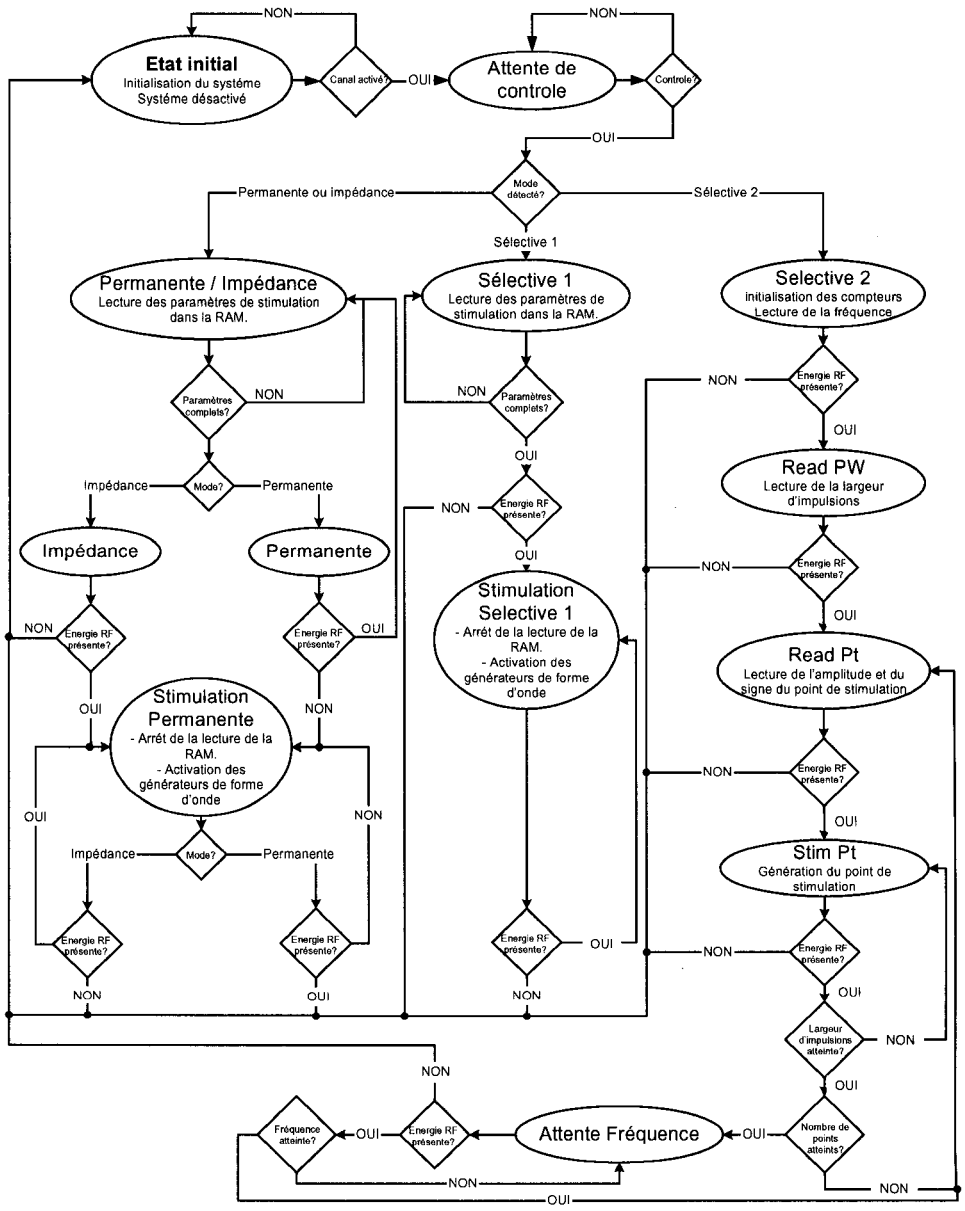


Figure 4-7: Diagramme d'états du contrôleur MSA2 du générateur de stimuli

La MSA2 contrôle les adresses du démultiplexeur pour envoyer le bon paramètre au moment adéquat. Elle est aussi en charge, après avoir complétée les opérations de lecture, d'activer selon le mode de stimulation, les blocs de génération de forme d'ondes.

La Figure 4-8 présente le diagramme bloc du module de générateur de forme d'ondes SWG *(Stimuli Waveform Generator)* qui est utilisé pour les modes de stimulations sélective et permanente ainsi que pour la mesure d'impédance La fréquence d'entrée de 300KHz est tout d'abord divisée pour générer l'horloge du module TON/TOFF qui, comme son nom l'indique détecte les temps d'activation ou de désactivation de la stimulation permanente ainsi que ceux de calibration ou d'évaluation pour la mesure d'impédance. Deux autres diviseurs de fréquence sont utilisés pour les horloges de la haute et de la basse fréquence dans le mode sélectif. Ces horloges servent à la génération des impulsions hautes et basses fréquences à l'aide des modules PULSE GENERATOR. Ceux-ci reçoivent en entrée les valeurs de fréquences codées sur huit bits (HFP et LFP) et par une combinaison d'un compteur et d'un comparateur vont générer des signaux *Pulse_H* et *Pulse_L* chaque fois que l'une de ces fréquences sera atteinte. La fréquence d'horloge initiale de 300KHz est divisée par quatre pour la haute fréquence et par 256 pour la basse fréquence.

L'activation de ces modules est commandée par la MSA2 à l'aide du signal *Enable_SWG.* Une petite unité de contrôle est utilisée pour assurer la coordination des signaux à basse et haute fréquences. Celle-ci commande aussi les signaux de sélection de deux multiplexeurs (MUX). Le premier MUX sert à déterminer l'amplitude des signaux de sorties, tandis que le deuxième est utilisé pour la génération des largeurs d'impulsions

des stimuli. La sortie de ce multiplexeur est envoyée au moniteur de largeur d'impulsions WIDTH GENERATOR. Il produit deux signaux *UP et DOWN* qui vont être combinés pour générer un signal *ACTIVE*; celui-ci permet l'activation des stimuli. La polarité est indiquée par le signal *SIGN*. Un signal d'activité *(Active)* est renvoyé à l'unité de contrôle pour éviter tout chevauchement entre les stimuli à basse et à haute fréquence. Notons que la génération de signaux basse fréquence a la priorité sur celle de haute. Pour obtenir les largeurs d'impulsions adéquates, le compteur du module WIDTH GENERATOR compte le nombre de cycles d'horloge (300KHz) jusqu'à atteindre les valeurs HFW et LFW.

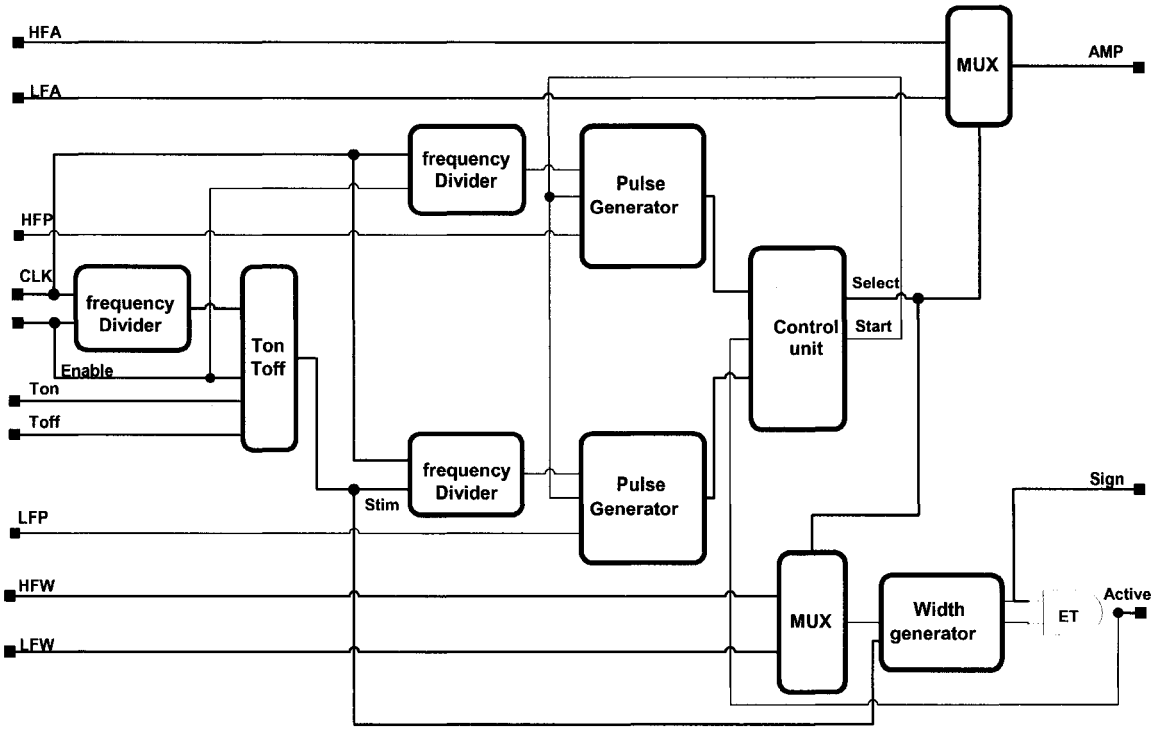


Figure 4-8: Diagramme bloc du module de génération de stimuli

Lors de l'utilisation du mode permanent ou celui de la mesure d'impédance, seuls le diviseur de fréquence et le bloc PULSE GENERATOR servant à la génération de la

basse fréquence sont utilisés. Cette simplification venant évidemment du fait que les stimuli basse fréquence de la stimulation sélective ont le même ordre de grandeur que la fréquence de stimulation des deux autres modes. Dans ces cas là, les deux blocs dédiés à la génération de la haute fréquence sont désactivés.

Dans le cas de la stimulation sélective flexible, la MSA2 participe au processus de génération de points. Le bloc Gen_PTS est dédié à la création du pattern de stimulation pour le mode de stimulation sélective flexible et son diagramme bloc est présenté à la Figure 4-9. Ce module reçoit en entrée l'amplitude du point (AMP [7:0]) sur huit bits, la durée d'impulsion (PW) à produire, la fréquence (FREQ) à laquelle le patron de stimulation sera répété ainsi qu'un signal d'activation *ENABLE* provenant de la MSA2 qui indique au bloc le début de la stimulation.

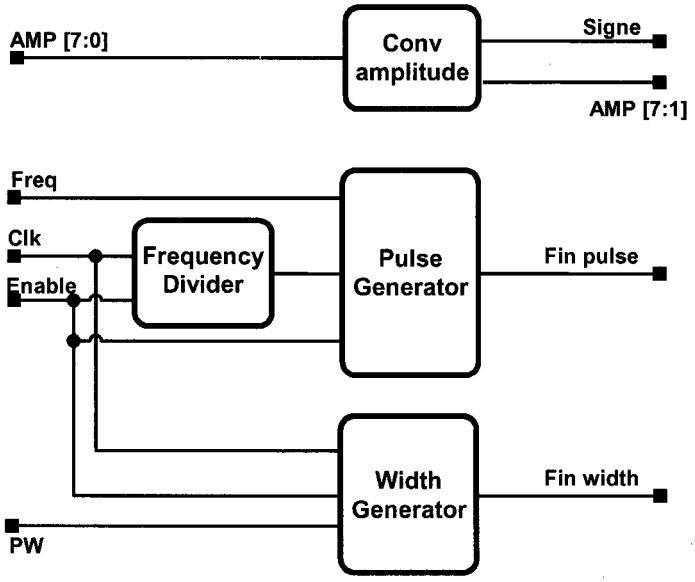


Figure 4-9 : Diagramme bloc du module de génération de points

Lorsque la durée d'impulsion souhaitée est atteinte, le bloc WIDTH GENERATOR produit un signal *Fin_width.* Lorsque ce signal est reçu par la MSA, elle

déclenche la génération d'un nouveau point. Si tous les points sont produits, le bloc se met en mode d'attente. Quand la valeur de fréquence du « pattern » est atteinte, la MSA reçoit un signal *Fin_pulse* et le cycle reprend. Le bloc CONV_AMPLITUDE est utilisé pour déduire du paramètre AMP l'amplitude et la polarité du point (signal *Signe*).

Les derniers blocs constituant le générateur de stimuli sont un MUX, une machine de contrôle (MSA_Z) et un bloc de conversion de signaux. Le MUX, commandée par la MSA2, est utilisé pour sélectionner, selon le mode de stimulation, les sorties du bloc SWG ou celles du module GEN_PTS.

MSA_Z est utilisée principalement pour la génération de stimuli dédiées à la mesure d'impédance. La Figure 4-10 décrit le diagramme d'états de la MSA_Z qui comporte quatre états principaux. Si le mode de mesure d'impédance est activé, la machine se met dans un état de calibration. Elle génère un signal *Test_Cal* qui est utilisé pour initialiser le bloc de mesure d'impédance. MSA_Z entre ensuite dans un état d'attente qui est interrompu par un signal signifiant la fin de la période de calibration T_setup. Survient alors une phase de mesure de l'impédance où le signal *Test_Z* est envoyé pendant un temps de mesure T_eval. Le cycle se termine enfin par un état d'attente où la valeur mesurée de l'impédance est transmise au contrôleur externe.

Lorsque le système est dans un autre mode de stimulation que la mesure d'impédance, la machine envoie un signal d'activité STIM au module SIG_TRANS. Ce bloc est un convertisseur de signaux: il est utilisé pour transformer les signaux de commande des modules SWG et GEN_PTS en signaux de contrôle pour la partie

analogique du canal. Ces signaux de contrôle sont des combinaisons logiques des signaux *ACTIVE, SIGNE, Test_Cal, Test_Z et STIM.*

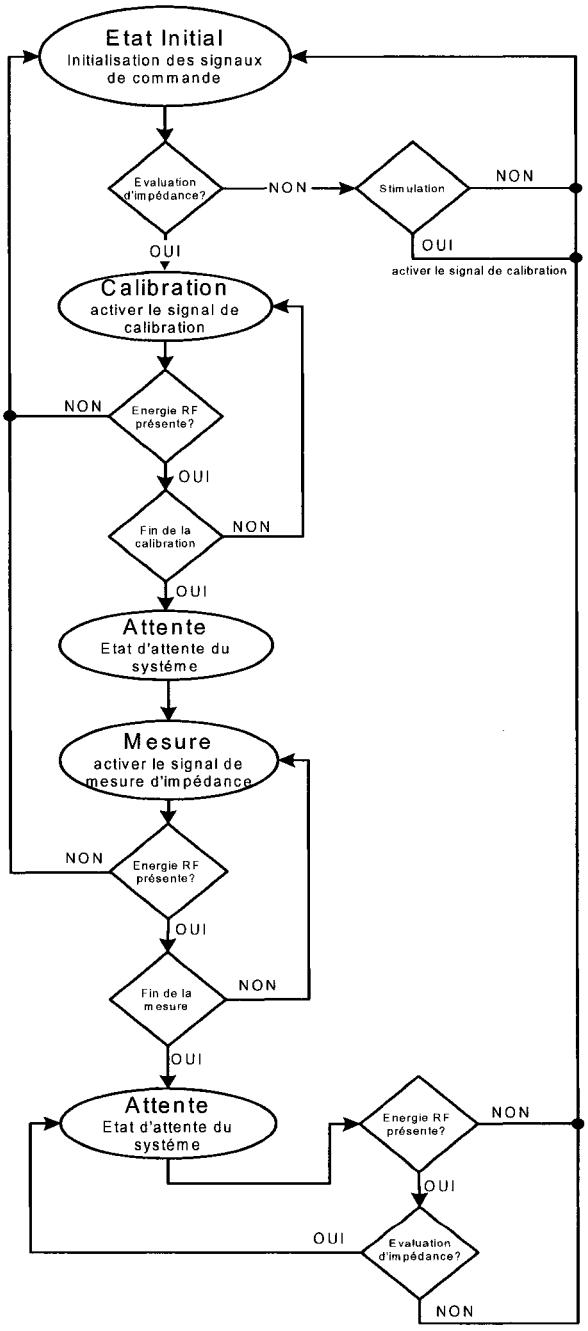


Figure 4-10: Diagramme d'états de la machine de conversion de signaux

Le signal *receiving* permet de déterminer la présence de l'énergie RF, car rappelons le, le système d'alternance d'alimentation, utilisé pour dissocier la stimulation permanente des autres types de stimulation, fait que le système est constamment alimenté. Les séquences des deux modes de stimulation sélective et de la mesure d'impédance ne se font que lors de la présence de l'énergie RF. Ainsi, dans ces modes d'opération, à chaque cycle d'horloge, le niveau logique de ce signal est vérifié et s'il est égale à un zéro logique (correspondant à aucun envoi d'énergie par le contrôleur externe), le système est automatiquement réinitialisé et attend les prochaines commandes de stimulation.

c) Stimulation en courant constant et mesure d'impédance

La Figure 4-11 représente le diagramme bloc de l'étage de sortie du canal de stimulation. Le design de la mesure d'impédance est inspiré des travaux de Donfack [16]. On peut y identifier un circuit de stimulation (SSC), un circuit de calibration (CSC), un convertisseur numérique à analogique (CNA), une source de courant et un oscillateur contrôlé en tension (VCO) utilisé pour la mesure d'impédance.

Le bloc SSC est utilisé pour transformer les signaux de contrôle provenant du module générateur de stimuli en courant qui sera injecté dans le nerf. Deux paires de transistors contrôlées par les signaux PS1, PS2, NS1 et NS2 agissent comme des commutateurs et leur configuration en H permet de transformer un courant constant en un courant biphasique balancé. Alors que les signaux NS1 et NS2 commandent les grilles de

transistors NMOS, les signaux PS1 et PS2 contrôlent des transistors PMOS. Le circuit SSC est aussi utilisé pour la mesure d'impédance.

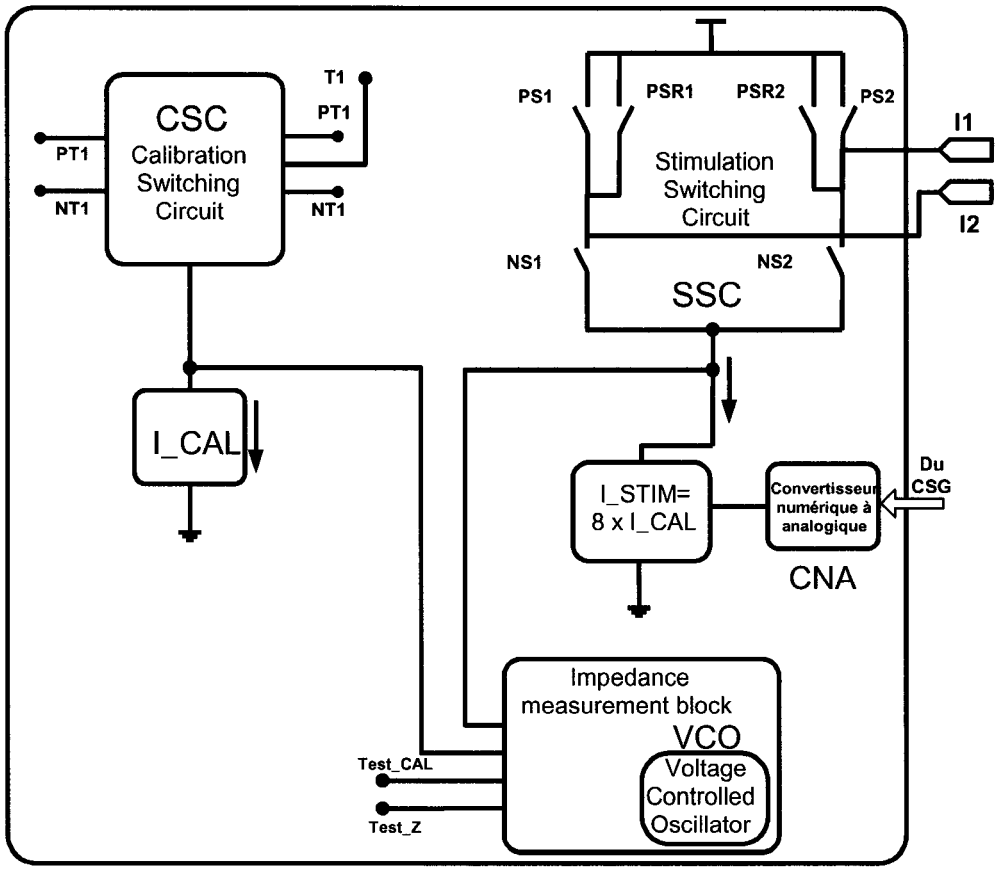


Figure 4-11: Diagramme bloc du canal de stimulation (adapté de [16])

Les transistors PMOS (PS1, PS2, PSR1, PSR2) utilisés peuvent subir une chute de tension substantielle, qui peut conduire à des décalages dans l'opération de conversion de la valeur d'impédance du nerf en tension. Ainsi durant le mode de mesure d'impédance, des transistors de type PMOS, mais plus résistifs, sont commandés par les signaux PSR1 et PSR2. La chute de tension de ces transistors est constante avec un courant de drain fixe et une polarisation constante, tandis que les transistors NMOS (NS1 et NS2) ont une faible chute de tension qui est moins critique pour la mesure d'impédance.

L'injection d'un courant dans le nerf génère une tension aux bornes des électrodes. Cette tension est envoyée à un oscillateur contrôlé en tension (VCO) dont la sortie est envoyée à un bloc d'estimation de fréquence. Celui-ci est composé de deux modules principaux : Un compteur huit bits qui évalue pendant une durée de temps fixe le nombre de cycles d'horloges produit par le VCO et une machine à états (MSA3) qui fixe la durée du temps de comptage. Le compteur est tout d'abord initialisé et la MSA3 est dans un état d'attente afin d'éviter toute erreur d'évaluation due à la période de stabilisation du VCO. Un état de comptage survient ensuite. La période fixe d'estimation est 2^k la période de l'horloge et ainsi la relation compte/fréquence est donnée par l'équation :

$$n = f_{osc} \times \frac{2^k}{f_{clk}} \qquad \textbf{(4.1)}$$

Avec *fosc* et *fclk* respectivement la fréquence à être convertie donnant une valeur relative de l'impédance et la fréquence d'horloge; n est le nombre de coups d'horloge obtenu après le compte et dans notre cas, k=8 (taille du compteur).

Un circuit de calibration (CSC) a été implémenté pour réduire les effets de variations de procédé ou de température. La même structure que celle du circuit de stimulation est utilisée. Deux transistors PMOS sont contrôlés par les signaux PT1 et PT2, tandis que la paire de transistors NMOS est contrôlée par les signaux NT1 et NT2. Les tissus stimulés (nerfs) sont remplacés par un transistor conçu pour opérer en mode triode. Commandé par le signal T1, il simule une résistance R qui permet au VCO de convertir une tension initialement connue **R x Itest** et ainsi de calibrer sa réponse.

Le courant injecté dans le nerf provient d'une source de courant réalisé à l'aide d'un CNA et d'un miroir de courant wide-swing. Le CNA est composé de sources de courant pondérées de façon binaire et commandées indépendamment par des bits d'activation provenant du générateur de stimuli (CSG). Les sources sont composées de transistors identiques de type P disposés en série ou en parallèle afin d'obtenir un courant plus ou moins élevé selon le nombre de transistors activés. Le fait d'utiliser à la fois des transistors en série pour la moitié des sources et d'autres en parallèle pour l'autre moitié permet de réduire la surface occupée par le CNA. La valeur du courant de référence appliquée au miroir de courant est la somme des courants produits par toutes les sources actives.

4.3 Réalisation du circuit intégré

Tel que nous venons de le démontrer, le design proposé est de nature mixte. Il combine des blocs analogiques à une partie de contrôle numérique. Nous allons décrire la réalisation des différents blocs constituants le circuit et ensuite présenter la technique utilisée pour l'interconnexion de ces blocs.

4.3.1 Bloc numérique

Le bloc numérique comprend la partie de contrôle du circuit. Nous y retrouvons, le contrôleur interne, les deux blocs générateurs de stimuli ainsi que la partie numérique de la mesure d'impédance; à savoir une machine à états (MSA5), un compteur ainsi que l'étage de sérialisation de la valeur d'impédance. Les différentes parties de ce bloc numérique ont été générées à l'aide du langage VHDL tandis que l'implémentation des

masques s'est faite par l'intermédiaire du guide de conception numérique de la Société Canadienne de Microélectronique *(CMC's Digital Design Flow)*.

La Figure 4-12 montre le ''layout'' complet du bloc numérique. Nous y voyons un module, connecté aux plots d'entrée/sorties, prêt à être envoyé pour fabrication. Nous avons du modifier ce résultat pour l'adapter à notre design. Nous expliquerons ces modifications en détails un peu plus loin dans ce chapitre.

4.3.2 Étage analogique de sortie

L'étage de sortie comprend le canal de stimulation ainsi que le VCO utilisé pour la conversion en fréquence de la mesure d'impédance. Le bloc avait été réalisé par Donfack [16] et était fonctionnel dans la technologie CMOS 0.35um.

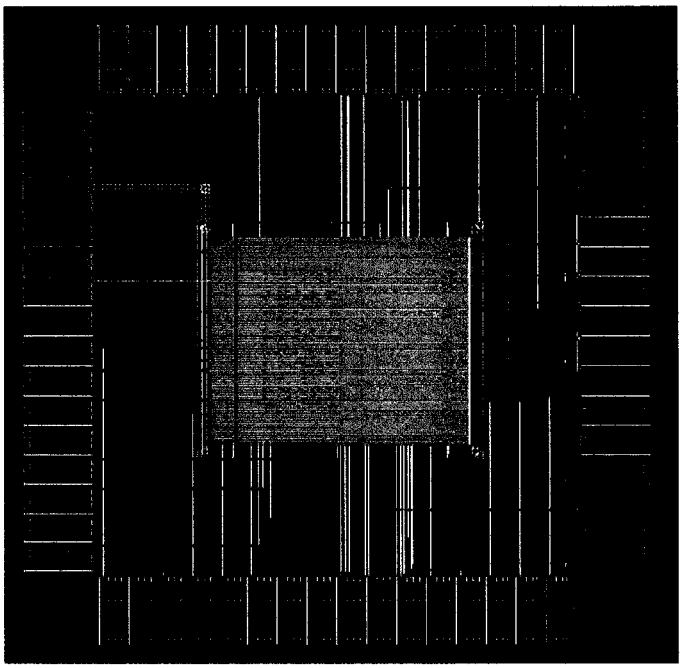


Figure 4-12: Dessin de masques du bloc numérique (900x710 um^2)

Nous avons ainsi effectué une conversion de cet étage de sortie de la technologie CMOS 0.35um à la technologie CMOS 0.18um pour être en mesure de le fusionner aux autres parties de l'implant proposé.

Après avoir vérifié le fonctionnement du design par des simulations avec le logiciel *Analog Artist,* nous l'avons exporté en technologie 0.35u. Nous l'avons ensuite importé dans le logiciel de génération de layout dans la technologie CMOS 0.18u. Nous avons adapté les valeurs de dimensions des différents éléments actifs en raison du changement de technologie. De plus, l'absence de correspondance de certaines couches de métaux dans les deux technologies nous a mené à créer une table de correspondance pour le remplacement de ces couches. Après la conversion du design en technologie CMOS 0.18um, nous avons procédé à l'ajustement des tailles minimales des contacts, des transistors et respecté les valeurs d'espacements dans la création des différents éléments constitutifs du design. La Figure 4-13 présente les masques de l'étage de sortie analogique. On peut identifier dans le coin inférieur gauche le VCO de la mesure d'impédance avec la capacité associée, tandis que dans le coin inférieur droit, on retrouve le convertisseur numérique à analogique et les miroirs de courant. La partie supérieure du layout contient les circuits de stimulation et de calibration SSC et CSC.

La simulation du bloc de conversion numérique à analogique nous a permis de réaliser que la valeur de courant maximale fournie n'était pas suffisante pour assurer la valeur 2mA requise pour notre application après le passage par le miroir de courant.

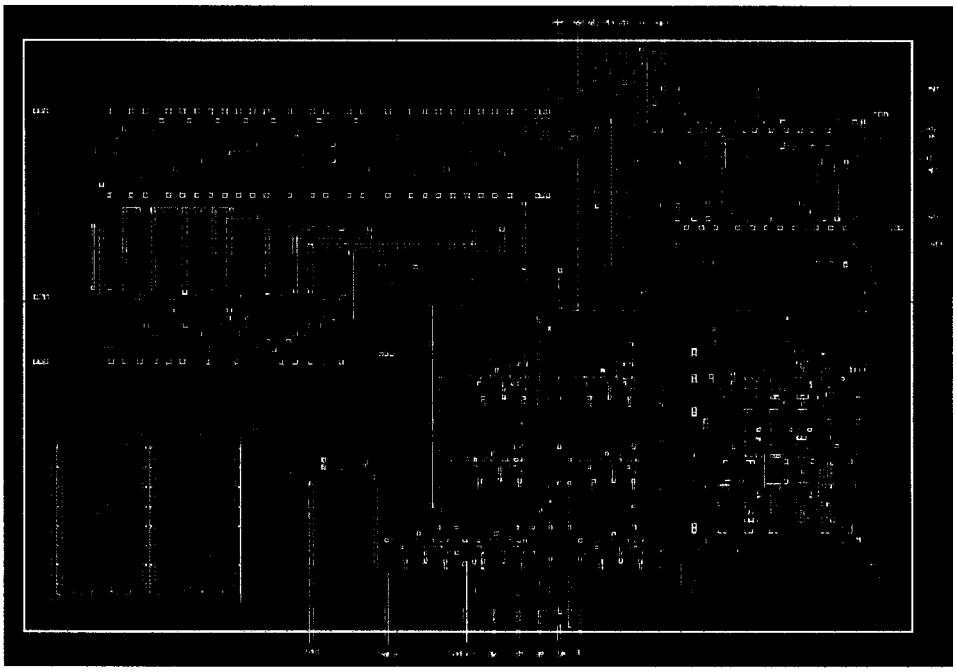


Figure 4-13: Layout de l'étage analogique de sortie (200x150 um^2)

Nous avons ainsi procédé à un ajustement au niveau des transistors composant le CNA. Nous avons augmenté les largeurs des canaux des transistors pour augmenter le courant fourni. En effet, les relations définissant la relation Courant-Tension pour un transistor MOS idéal sont données par les équations (4.2) et (4.3) en mode triode et région active respectivement:

$$I_d = \mu_n . C_{ox} . \frac{W}{L} . V_{eff} . V_{ds} \qquad \textbf{(4.2)}$$

et

$$I_d = \frac{\mu_n . C_{ox}}{2} . \frac{W}{L} . V_{eff}^{\ 2} \qquad \textbf{(4.3)}$$

où $V_{eff} = V_{gs} - V_t$

Assumons que V_{eff} et V_{ds} sont fixées, le courant serait proportionnel au rapport de la largeur sur la longueur.

La tension d'alimentation nominale de la technologie CMOS 0.18um est de 1.8V, et elle peut atteindre 3.3V. Cependant, pour injecter un courant de 2mA dans le nerf, comme nous l'avions démontré dans le chapitre II, il est nécessaire d'avoir au moins une tension d'alimentation de 5V. L'utilisation des transistors réalisés en CMOS 0.18um avec une tension de 5V engendre des conséquences pouvant être néfastes. En effet, les transistors avec une courte longueur de canal et un grand champ électrique appliqué à la grille peuvent subir des dégradations dans la mobilité effective de leur porteurs dues à plusieurs facteurs. Le champ électrique élevé force la profondeur effective du canal à changer et cause aussi plus de collision d'électrons et ainsi réduit leur mobilité. Un autre facteur causant cette dégradation est que, du encore une fois au grand champ électrique, la vélocité des porteurs commence à saturer. Cette saturation force la relation courant-tension initialement de type loi-carrée à devenir imprécise. La nouvelle relation varie entre une relation linéaire et une loi carrée. Dans la plupart des circuits de conversion tension-courant, qui se base sur cette caractéristique (loi carrée), cette imprécision peut-être une source majeure d'erreur. Des longueurs de canal supérieures à la taille minimale permettent de minimiser ces dégradations.

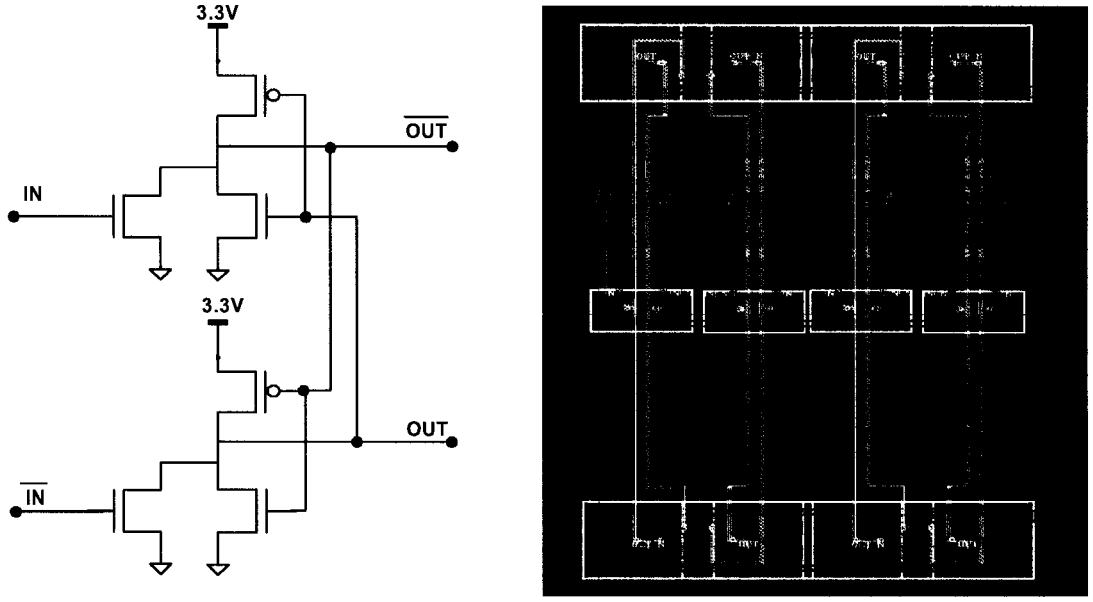
Un autre problème peut être dû aux «Hot carriers». Ces porteurs à très grande vélocité peuvent causer des effets nuisibles tels que la génération de paires trous-électrons par ionisation d'impact ou effet d'avalanche. Ces paires vont causer le passage d'un grand courant entre le drain et le substrat. Ce flot de courant peut causer des grandes

chutes de tensions à travers le substrat et possiblement des ''latch-ups''. De plus une dégradation rapide des paramètres du composant (Vth, gm, Rds...) est à craindre.

Ainsi pour pouvoir appliquer la tension d'alimentation de 5V, nous avons pris des valeurs de longueurs de transistors relativement grandes afin de réduire les effets du champ électrique élevé.

4.3.3 Interconnexion en partie numérique et analogique

Après avoir réalisé nos blocs numériques et analogiques, l'étape suivante était de les relier physiquement. Le problème qui se posait était que le bloc numérique est alimenté avec une tension nominale de 1.8V, tandis que la partie analogique est reliée à une alimentation nominale de 3.3V, mais qui dans le cas de notre application peut aller jusqu'à 5V. Nous avons ainsi utilisé des blocs de changement de niveau : «*Level Shifters*». Ces blocs, dont le schéma de fonctionnement et le « layout » sont présentés à la Figure 4-14, transforment les signaux d'entrées à 1.8V en des signaux de sorties à une tension égale à la tension d'alimentation VDD. La liaison entre les blocs analogiques et numériques a été faite de façon manuelle en intercalant un de ces blocs entre chaque sortie numérique et son entrée correspondante analogique.



(a) (b)

Figure 4-14: Bloc de changement de niveau 1.8V-3.3V : a) Schéma ; b) Layout (40x50 um^2)

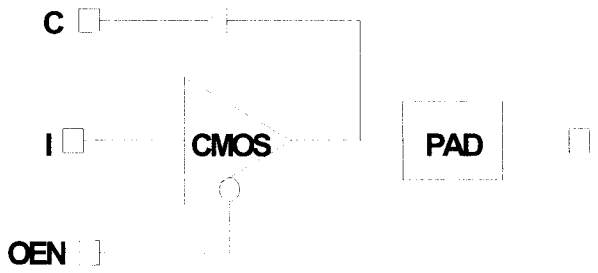
4.3.4 Plots de sortie et testabilité

Au niveau des entrées/sorties de la puce, nous avons utilisé quatre types de plots. Nous avons utilisé les PDIDGZ pour les entrées, tandis que pour les sorties, nous nous sommes servis des PDO08CDG. Ces deux types de plots proviennent de la librairie *TPZ973G* de la technologie TSMC 0.18um. Pour nos sorties analogiques, notamment celles qui délivrent du courant aux nerfs, les plots analogiques PANALOGESD ont été utilisés. La majeure particularité provient de l'utilisation de plots bi-directionnel PDB04DGZ pour la testabilité de notre circuit. Ces plots, dont le schéma de fonctionnement et la table de vérité sont présentés à la Figure 4-15 permettent d'observer la valeur présentée par le signal I à l'extérieur de la puce sur le port PAD. Selon le niveau

logique imposé à l'entrée OEN, le port PAD est transformé en un port d'entrée et toute valeur qui y sera imposée sera transmise au signal C. Nous avons utilisé ces plots pour un des canaux de stimulation en les intercalant entre le bloc numérique et les blocs analogiques. Ils nous permettront ainsi de tester le fonctionnement du circuit en vérifiant que les bonnes valeurs sont transmises par le bloc numérique à l'étage de sortie du stimulateur. Ils permettront ainsi d'isoler le bloc numérique de la partie analogique et de fixer des valeurs de test pour la vérification de la partie analogique.

Truth Table			
INPUT			OUTPUT
OEN	I	PAD	C
1	x	0	0
1	x	1	1
1	x	Z	x
0	0	0	0
0	1	1	1

(a)



(b)

Figure 4-15: (a) Table de vérité ; (b) Schéma illustratif des plots PDB04DGZ (tirés de la CMC)

La Figure 4-16 présente le ''layout'' final de la puce. On peut y identifier tous les blocs décrits précédemment inter-reliés.

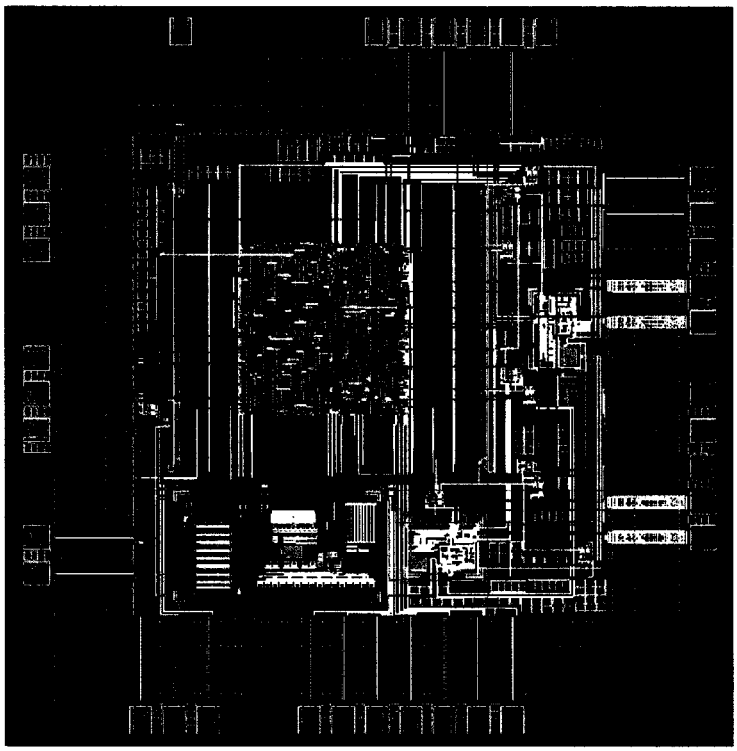


Figure 4-16 : ''Layout'' de la puce dédiée (2000x2000 um^2)

4.4 Conclusion

Dans ce chapitre, nous avons présenté l'architecture de notre nouveau neurostimulateur intégré ainsi que les étapes qui ont conduit à sa réalisation. Il est en mesure de générer plusieurs types de stimulation. Grâce à la flexibilité apportée par la reprogrammabilité des paramètres de stimulation ainsi que sa reconfigurabilité, il remédie aux différents manquements présentés par les systèmes de stimulation antérieurs.

CHAPITRE 5
VALIDATION DES SYSTÉMES DE STIMULATION ET EXPÉRIMENTATIONS EN LABORATOIRES.

Nous présentons dans ce chapitre un bilan des travaux effectués tout au long de notre maîtrise et nous apportons un complément aux résultats présentés dans l'article du chapitre 3.

5.1 Système de stimulations sélective et permanente

Dans les chapitres précédents de ce mémoire, nous avons présenté les travaux de conception et de réalisation d'un stimulateur implantable. Après avoir identifié les modifications nécessaires à apporter à la version de stimulateur précédemment réalisé par notre équipe de recherche, nous avons tout d'abord effectué une validation en laboratoire du système de simulation sur une plaquette d'interconnexions. Cette étape de validation est décrite dans le chapitre 2. Elle a permis d'identifier quelques défauts de fonctionnement. Après la correction de ces derniers et le montage des composants en surface sur un circuit imprimé, les implants ont été moulés à l'aide de l'époxy de la compagnie *Epoxy technologie* présentée au chapitre 2. La dernière étape dans la préparation des implants a consisté à les recouvrir de silicone biomédicale, afin de protéger les tissus biologiques avec lesquels il sera mis en contact. Sur la photographie de

la Figure 5-1, on peut voir un stimulateur enrobé de silicone transparent avec les deux connecteurs où viendront se brancher les électrodes de stimulation.

Un contrôleur externe de type CTF plus robuste et résistant a aussi été réalisé. Assemblé en un seul bloc, tout risque de rupture de communication entre la partie de modulation et de transmission est supprimé. Les modifications apportées aux boutons poussoirs permettent aussi une manipulation plus aisée par les usagers.

Figure 5-1: Photographie du microstimulateur prêt à être implanté.

Le système comprenant le stimulateur implantable ainsi que son contrôleur externe a fait l'objet d'évaluation dans un contexte réel d'utilisation lors d'expérimentations animales. Plusieurs chiens paraplégiques ont été implantés et ont fait l'objet d'études en phase chronique grâce à une collaboration avec le département d'urologie et le centre des ressources animales de l'Université.

L'efficacité de stimulation sélective par blocage haute fréquence ainsi que les effets bénéfiques d'une stimulation permanente sur les problèmes d'hyperréflexie de la

vessie à la suite d'une lésion de la moelle épinière avaient déjà été démontré par notre équipe de recherche. Nous voulions ainsi plutôt évaluer la fiabilité et la durabilité du nouveau système de stimulations sélective et permanente dans un contexte réel d'utilisation à la suite des modifications apportées.

5.1.1 Protocole expérimental

Nous présentons tout d'abord le protocole chirurgical suivi par les dispositifs d'implantation du système. L'animal choisi pour l'étude est d'abord examiné afin de s'assurer que son système urinaire est intact et ne présente pas de malformations. La chirurgie a lieu après une semaine d'observation. L'animal est anesthésié et maintenu sous respirateur pendant l'opération alors que ses fonctions vitales sont continuellement surveillées par de l'appareillage spécialisé. De plus, les pressions vésicale et urétrale sont enregistrées par un analyseur urodynamique connecté à un ordinateur de type PC.

Une première incision au niveau de la vertèbre T10, afin de sectionner directement la moelle épinière, provoque la paraplégie. Ensuite une seconde incision est faite au bas de la colonne vertébrale au niveau de la racine sacrée. Les deux nerfs sacrés S2 sont alors identifiés puis stimulés à l'aide d'une sonde et d'un stimulateur externe intégrant le même étage de stimulation que le stimulateur implantable, mais avec un contrôle direct des paramètres. Une électrode est alors enroulée autour du nerf sacré S2 qui provoque la plus grande contraction de la vessie. Le stimulateur est ensuite connecté à l'électrode, puis glissé sous la peau entre l'hypoderme et les muscles. Le système de stimulation est testé une dernière fois avant de suturer l'incision. L'animal est placé dans une sorte de hamac pendant 3 ou 4 jours le temps qu'il s'habitue à sa nouvelle condition.

L'étude en phase chronique s'étale sur une période de 8 mois. Immédiatement après la chirurgie et pendant une période variable, pouvant atteindre 2 mois, la vessie entre dans une phase d'aréflexie (Absence de réflexes qui peut être totale ou partielle) durant laquelle aucune stimulation électrique ne sera appliquée. Elle est simplement vidée de façon conventionnelle au moyen d'un cathéter placé dans l'urètre. Lorsque débute la phase d'hyperréflexie, la stimulation permanente est activée et la vessie de l'animal est vidée deux fois par jour au moyen de la stimulation sélective. Lors de la première stimulation, un ensemble de paramètres de stimulation est choisi pour chaque animal afin de maximiser le volume évacué. Les techniciens animaliers s'assurent donc quotidiennement du fonctionnement de la stimulation sélective et en évaluent l'efficacité. Si le volume d'urine résiduel après une stimulation est supérieur à 50% du volume total d'urine contenu dans la vessie, mais que des contractions sont toujours perceptibles, les paramètres de stimulation devront être changés par le chirurgien. Toutefois, si aucune contraction musculaire n'est ressentie, une intervention chirurgicale est généralement requise, car cela peut signifier une défaillance du système de stimulation implanté.

Pour détecter le début et la fin des phases d'aréflexie et d'hyperréflexie, des cystomyogrammes (CMG) assistés par ordinateur sont effectués toutes les 2 semaines et permettent d'observer les pressions dans la vessie et dans l'urètre ainsi que l'activité électrique musculaire du detrusor. Une faible activité électrique musculaire permanente et une variation de la pression vésicale périodique sont caractéristiques de l'hyperréflexie. Ces CMG permettent aussi de contrôler plus précisément l'efficacité des paramètres de

stimulation sélective et sont le seul moyen de vérifier de l'extérieur le fonctionnement de la stimulation permanente.

Les exemples de CMG présentés aux Figure 5-2 et Figure 5-3 illustrent les variations de la pression et de l'activité électrique musculaire de la vessie enregistrées lors d'une stimulation sélective et lors d'une stimulation permanente respectivement.

Une stimulation sélective se caractérise par une très forte activité électrique musculaire (EMG) du detrusor, une augmentation simultanée de la pression vésicale (Pves) et urétrale (Pura), suivie d'une diminution de la pression urétrale liée au relâchement du sphincter qui provoque finalement la miction. Lors d'une stimulation permanente, on constate une activité électrique beaucoup plus faible (puisque les courants de stimulation sont réduits) et de légères contractions du detrusor et du sphincter suffisamment faibles pour ne pas causer de miction.

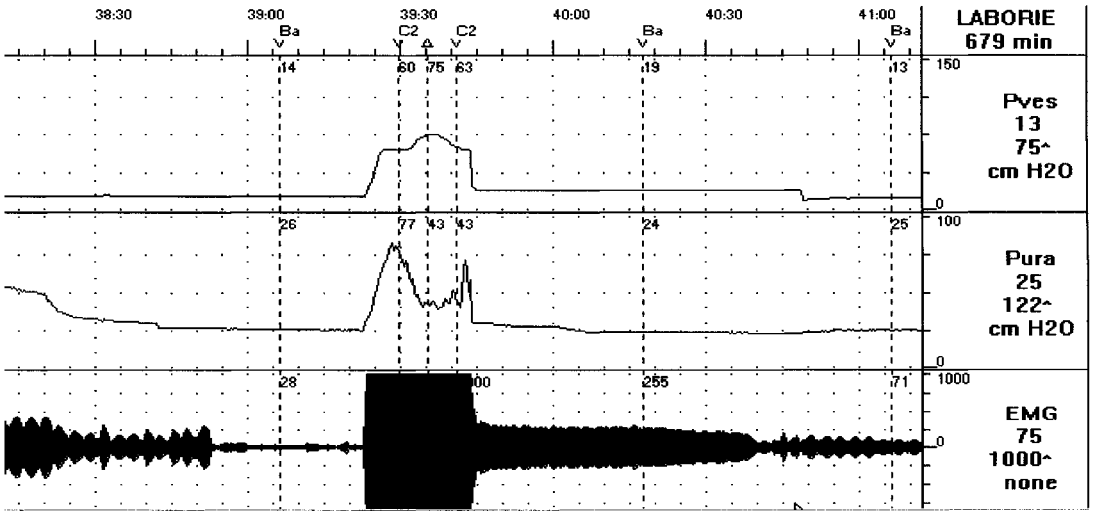


Figure 5-2 : Pression dans la vessie (Pves) et dans l'urètre (Pura) lors d'une stimulation sélective.

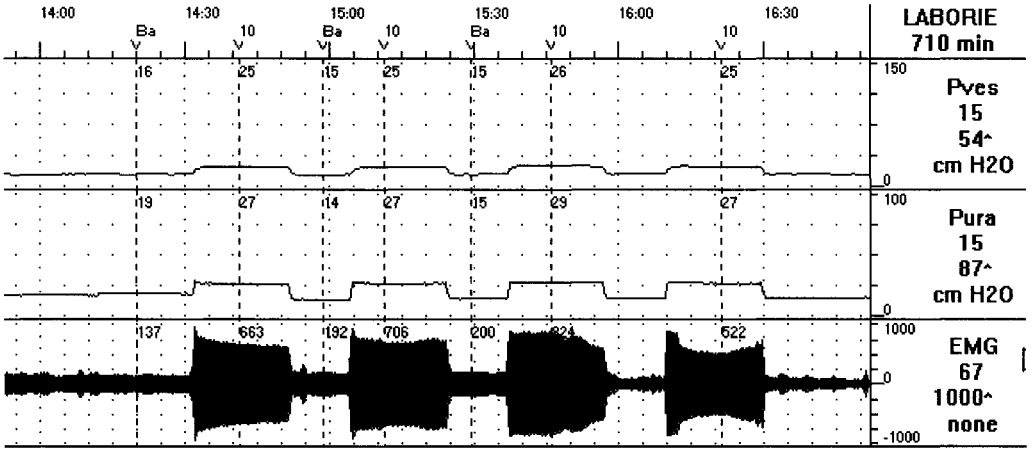


Figure 5-3 : Pression dans la vessie (Pves) et dans l'urètre (Pura) lors d'une stimulation permanente

5.1.2 Résultats

Nous présentons les résultats expérimentaux obtenus après les études chroniques sur quatre chiens à l'animalerie de l'Université McGill.

Le premier chien implanté, Deimo, a présenté des symptômes de l'hyperreflexie et la stimulation permanente a été utilisée pour réduire et supprimer les effets. Lors des trois premières semaines, l'application de la stimulation sélective a permis d'obtenir un volume d'évacuation d'environ 20%. Ce volume a ensuite été amélioré jusqu'à atteindre une valeur de 70% du volume total de la vessie lors des quatre semaines suivantes.

Lors des cinq premières semaines, la stimulation sélective produisait de très bonnes contractions chez le deuxième chien, Zam, sans produire d'évacuation. Cependant, le volume évacué a pu être augmenté à 60% après cette période de temps. Les mêmes bon résultats ont été obtenus chez le troisième chien Lyr, mais sur une période de 3 semaines.

Les résultats ont été moins concluants chez le quatrième chien, Dongo. Lors de l'opération, la réaction des nerfs à la stimulation a été difficile à produire. Après l'opération, de bonnes contractions de la vessie ont pu être obtenues. Au bout de quatre jours cependant, celles-ci ont disparu; l'électrode a dû être retirée et une nouvelle a été placée sur le deuxième nerf S2 sans pour autant produire de meilleures résultats.

Ces expérimentations ont été effectuées sur une période de cinq mois. Cependant, lors des derniers mois, celles-ci ont été caractérisées par plusieurs défaillances au niveau du matériel de stimulation (Interruption de fonctionnement de l'implant, absence de contractions, bris du contrôleur,...). Plusieurs démarches ont été entreprises pour améliorer la situation et à l'heure actuelle, un nouveau stimulateur est implanté sur un nouveau chien à une fréquence de trois à quatre semaines. Le Tableau 5.1 résume les différents résultats obtenus lors de l'étude en phase chronique chez les animaux.

Tableau 5.1 :Résumé des résultats expérimentaux obtenus chez 4 chiens à l'animalerie de l'université McGill.

Nom	Date Début	Date d'arrêt	Commentaires
Deimo	03/03/03	15/06/03	Symptômes d'hyperreflexie – utilisation de la Stim Permanente 20% évacuation les 3 premières semaines Améliorer à 70% les 4 semaines suivantes
Zam	10/03/03	11/08/03	Bonnes contractions sans évacuation pendant 5 semaines Amélioration jusqu'à 60 %
Lyr	31/03/03	15/07/03	60% évacuation au bout de 3 semaines
Dongo	12/05/03	10/06/03	Bonnes contractions les 4 jours suivant l'opération. Changement d'électrode – utilisation du deuxième nerf S2

5.2 Neurostimulateur programmable intégré

Le nouveau système intégré a été conçu pour répondre au besoin croissant de miniaturisation et de minimisation de la consommation d'énergie du système présenté à la section précédente. Son architecture détaillée ainsi que son fonctionnement général ont été présentés dans le chapitre 4. Le circuit intégré a été réalisé dans la technologie CMOS 0.18um par l'intermédiaire du service de fabrication de la Société Canadienne de Microélectronique (SCM). Avant d'envoyer le circuit pour fins de fabrication, nous l'avons soumis à un cycle de stimulations exhaustives pour en vérifier le fonctionnement théorique.

Nous rapportons ainsi quelques résultats obtenus lors de ces simulations et ensuite nous présentons les résultats obtenus lors des essais en laboratoire sur les échantillons obtenus de la SCM.

5.2.1 Résultats de simulation

La Figure 5-4 montre les résultats de simulation du bloc numérique DATA RECOVERY. Il décrit un exemple de réception d'une trame de stimulation valide. Nous pouvons voir la détection de l'entête dans le signal de données de la trame DATA. Après que six « 1 » aient été détectés, le signal SIX_ONE est activé. Le signal VALID indique la détection d'une commande de stimulation valide dans un canal inactivé (MODE_OUT et CHANNEL_OUT) et active le début du chargement des paramètres de stimulation dans le registre à décalage. Ces paramètres sont finalement sauvegardés dans la RAM du canal de stimulation indiqué. Cette opération est représentée dans le graphique par le signal WRITE qui commandent les opérations d'écriture dans la RAM. À la fin du

chargement le CRC est vérifié et s'il est validé, le signal CRC_OK est activé et la commande est donnée au canal de stimulation par le signal TAKE_CTRL. A la Figure 5-5, nous présentons les résultats de simulation du bloc numérique complet. On peut y identifier les signaux de stimulation (NS1, NS2, PS1,...,PT2). Trois types de stimulation sont présentés : une stimulation sélective flexible, une stimulation sélective classique ainsi qu'une mesure d'impédance. Les signaux de stimulation PS, NS et PT sont activés en alternance selon le mode de stimulation. Nous pouvons aussi identifier la valeur de l'amplitude (AMPLITUDE)des signaux de sortie qui varie. Comme dans le cas précédent de simulation, les signaux VALID et CRC_OK montrent respectivement la détection de commandes valides et la vérification de l'intégrité des données de stimulation.

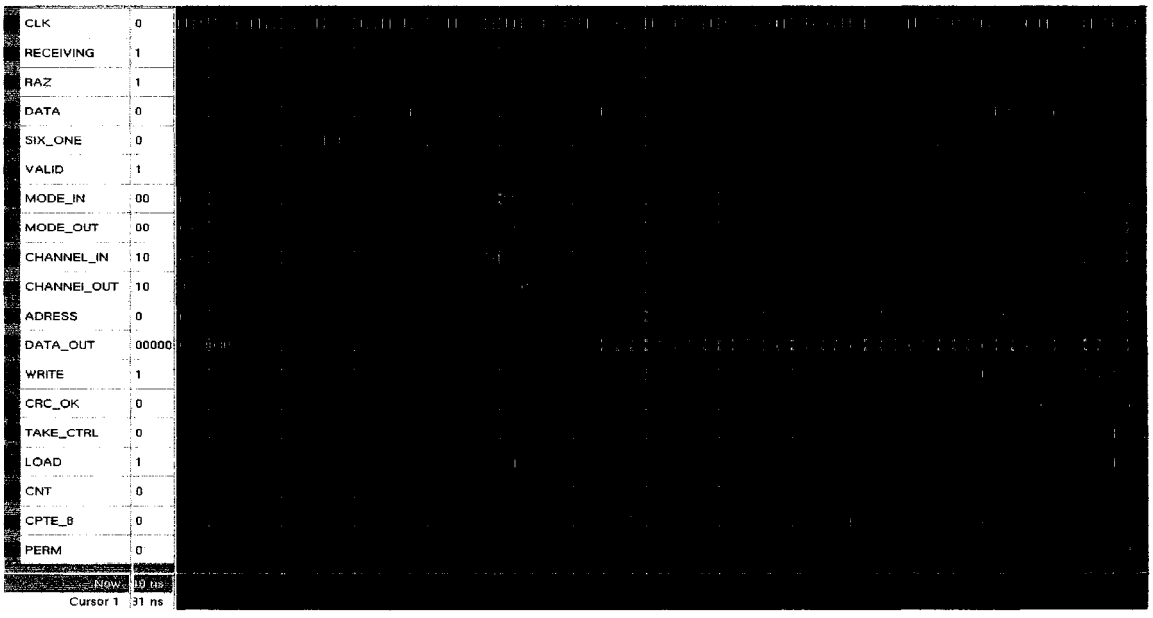


Figure 5-4: Résultats de simulation du bloc DATA RECOVERY : détection d'une trame de stimulation valide, transfert des paramétres et activation du bloc générateur de stimuli.

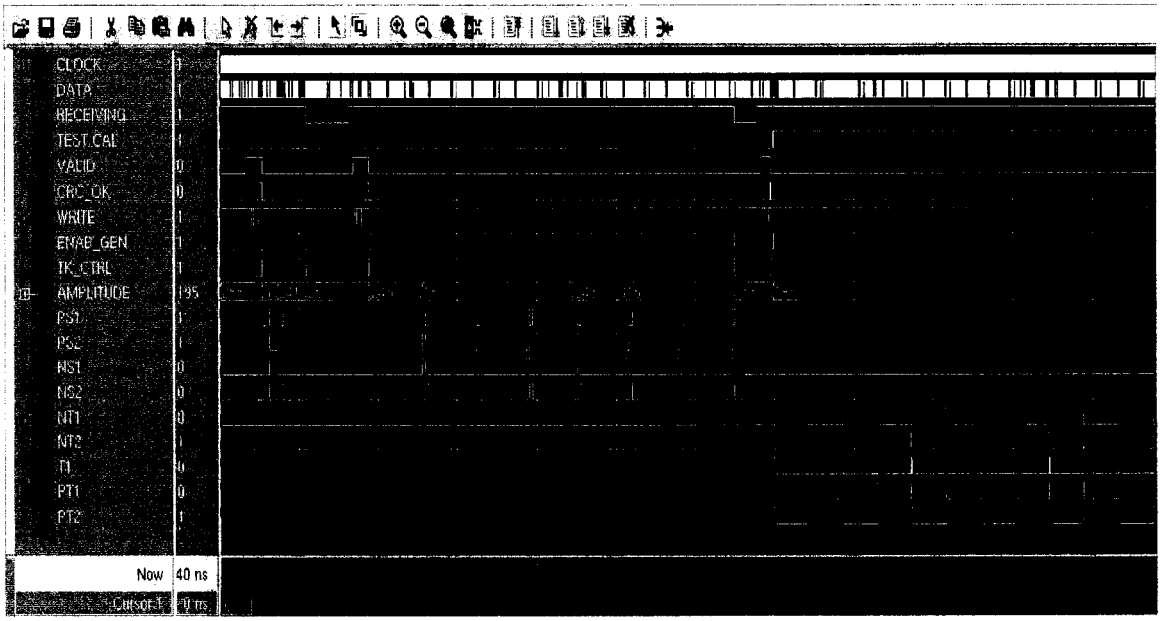


Figure 5-5 : Résultats de simulation du bloc numérique avec identification des signaux de commande de l'étage de sortie.

5.2.2 Résultats expérimentaux

Nous présentons dans cette section les résultats expérimentaux obtenus avec les cinq échantillons fabriqués par la compagnie TSMC et obtenus de la SCM.

Nous avons utilisé les fonctions de génération de trames de l'analyseur logique TLA715 de Tektronic pour produire les trames de stimulation dédiées aux tests de puce. Des résistances de 1KΩ ont été utilisées pour représenter l'impédance du nerf. Nous avons utilisé les fonctions de testabilité que nous avions intégrées dans le design pour caractériser de façon hiérarchique le fonctionnement de la puce. A l'aide des plots bidirectionnels, nous avons pu observer tous les signaux intermédiaires ainsi que ceux utilisés pour la communication entre le bloc numérique et la partie analogique. La Figure 5-6 présente la récupération de la trame de stimulation à partir du signal encodé Manchester. On peut identifier au début de la trame les 6 « 1 » composant l'entête. La

Figure 5-7, quant à elle décrit le transfert des paramètres de stimulation dans la RAM. Le signal du haut présente la trame de stimulation, tandis que l'autre signal montre le signal d'activation de l'écriture en RAM (Wr_N). Ce signal est activé tous les huit coups d'horloge afin de sauvegarder les huit bits du paramètre dans la mémoire.

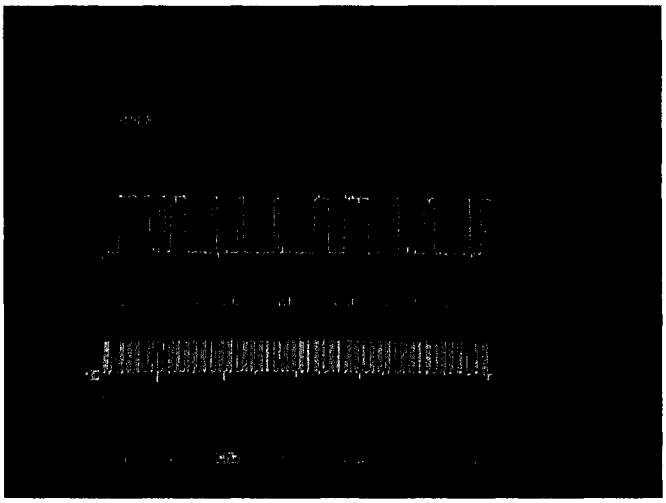


Figure 5-6: Résultats expérimentaux : décodage après réception de la trame des données de stimulation: le signal du haut est la trame de stimulation tandis que celui du bas montre le signal encodé Manchester.

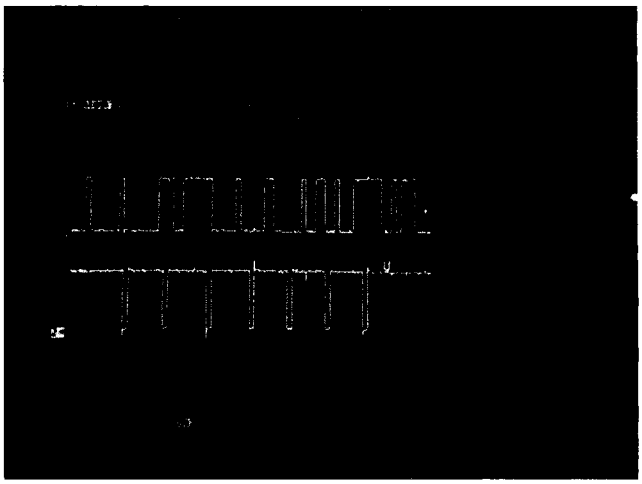


Figure 5-7: Résultats expérimentaux: écriture des paramètres de stimulation dans la RAM : le signal du haut présente la trame de stimulation, tandis que celui du bas montre le signal d'activation de l'écriture en RAM (Wr_N).

Nous avons ainsi pu, de cette manière, caractériser le comportement complet du bloc numérique et vérifier son fonctionnement prévu; cela notamment grâce à l'accessibilité aux signaux *Valid, CRC_OK*, ainsi qu'aux signaux de commandes des transistors de l'étage de sortie.

Lors de la vérification du bloc analogique, nous avons remarqué quelques dysfonctions. Nous avons initialement soupçonné les liaisons entre les blocs numériques et analogiques d'en être la cause. Après avoir appliqué de l'extérieur, toujours à l'aide des broches bidirectionnelles de test des signaux de commandes et obtenu le même comportement, nous avons finalement identifié la source du problème comme étant relié à l'entrée *Vref* du DAC. En effet, celle-ci est restée flottante et de ce fait cause une impossibilité de varier l'amplitude du courant de sortie. Nous avons pu cependant vérifier que les fonctionnalités globales du système ainsi que les valeurs des paramètres appliqués étaient respectées. Nous n'avons malheureusement pas pu caractériser la fonction de mesure d'impédance car celle-ci dépend directement de la valeur de courant injectée dans le nerf. Un nouveau circuit intégré est cependant en cours de fabrication pour corriger cette erreur et permettre de valider cette fonctionnalité.

Les Figure 5-8 et Figure 5-9 présentent des formes d'ondes de stimulation obtenus avec le système global. On peut y identifier respectivement une stimulation sélective ainsi qu'une stimulation flexible avec un patron de stimulation arbitraire. Au niveau de la stimulation sélective, nous identifions très bien la basse et la haute fréquence, et nous voyons que nous pouvons varier les durées d'impulsions. La stimulation permanente a été aussi validée, et les temps d'activation et d'inactivation *T_ON et T_OFF* étaient aussi

respectés. Au niveau des plages de variation, la haute fréquence (HF) varie de 294Hz à 75KHz tandis que la basse (BF) varie de 4.6Hz à 1.2KHz.Les durées d'impulsions générées vont de 3us à 853us.

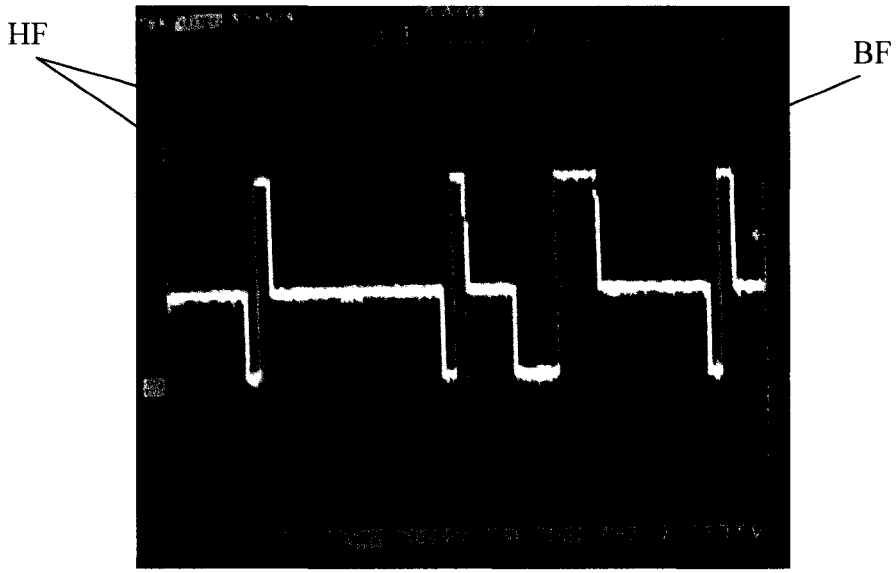


Figure 5-8: Résultats expérimentaux : présentation d'un signal de stimulation sélective avec identification de la haute (HF) et la basse fréquence (BF).

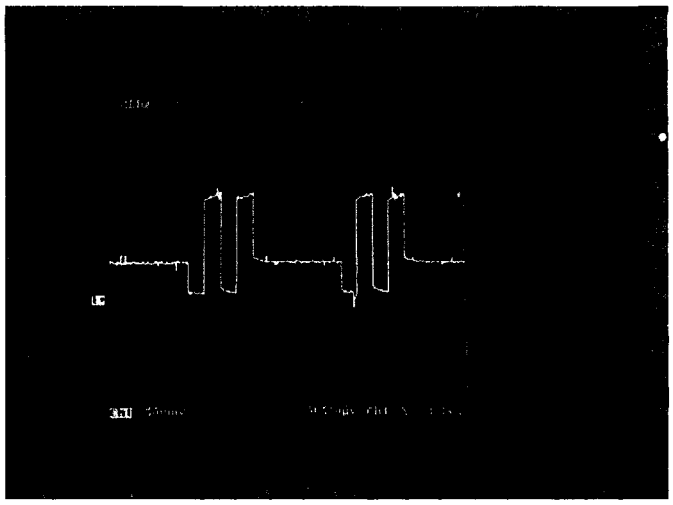


Figure 5-9: Résultats expérimentaux : présentation d'un signal de stimulation flexible.

DISCUSSION GÉNÉRALE

Lors de nos travaux, nous avons pu concevoir et tester un stimulateur sur un circuit imprimé. Nous avons pu valider son comportement en phase chronique et découvrir les exigences reliées à la réalisation d'expériences in vivo. Lors de la conception du nouveau contrôleur pour les expériences chroniques, nous avons pu réaliser l'importance d'un matériel de qualité et à toute épreuve dans le domaine de la stimulation électrique fonctionnelle. La réalisation d'un circuit intégré dédié nous a permis de comprendre et de maîtriser les notions et les contraintes associées à la conception de circuits miniaturisés. Nous avons pu faire face au défi posé par l'utilisation de tensions d'alimentation de niveaux différents dans le même système. De plus, nous avons pu réaliser les multiples possibilités offertes par les méthodes de conception de circuits et systèmes mixtes (analogique et numérique).

Les résultats obtenus à partir des tests en laboratoire et des expérimentations chez l'animal ont permis de confirmer les résultats précédemment rapportés par notre équipe de recherche: à savoir l'efficacité de la stimulation sélective par blocage à haute fréquence pour la miction volontaire chez les paraplégiques ainsi que la pertinence de la fonction de stimulation permanente pour le traitement de l'hyperréflexie. L'introduction de notre système intégré de stimulation permet de combler les manquements des systèmes de stimulation commerciaux et permet par la mesure d'impédance d'assurer un suivi permanent sur l'état des tissus stimulés lors d'implantation à long terme. Nous avons pu nous rendre compte de l'importance de celle-ci au regard des problèmes

rencontrés lors des opérations. Certaines interventions chirurgicales auraient pu être évitées si cette technique avait été disponible dans les versions précédentes.

CONCLUSION

Les travaux présentés dans ce mémoire sont dans la lignée des efforts déployés par notre équipe de recherche lors de la dernière décennie pour trouver et proposer des solutions aux problèmes rencontrés lors de la réadaptation du système urinaire. Nous avons présenté un système implantable utilisé pour le rétablissement des fonctions de la vessie.

Le présent mémoire contribue ainsi de façon importante à la recherche dans le domaine de la réadaptation du système urinaire. En effet, le nouveau stimulateur présenté permet aussi bien une miction volontaire sans effets secondaires qu'un traitement de l'hyperréflexie du muscle de la vessie; de plus elle fournit l'information nécessaire sur l'évolution de l'état du nerf stimulé lors d'une implantation de longue durée. La nouvelle stimulation flexible, quant à elle, permet aussi d'avoir plus de contrôle sur les stimulations appliquées. Elle assure une plus grande efficacité des méthodes de stimulation ainsi qu'une sécurité supplémentaire quand à l'injection de charges dans les tissus. De plus, l'intégration permet de réduire la surface utile ainsi que la consommation d'énergie du système. Cette nouvelle constitution de l'implant permet ainsi d'envisager des possibilités pour une validation chez l'humain dans un futur proche. Cependant plusieurs étapes de tests restent à être effectuées. Comme nous l'avons indiqué précédemment, un nouveau circuit intégré est en cours de fabrication et permettra d'effectuer la mesure d'impédance ainsi que de la stimulation flexible. Cette validation devra être faite lors d'expérimentations in vivo chez l'animal.

Recommandations

Afin que la technique de mesure d'impédance soit la plus précise et représentative possible, la mesure d'une impédance complexe devrait être incluse dans l'implant.

L'orientation vers des dispositifs portables plus évolués comme contrôleur portatif tels que les ordinateurs de poche (Palm pilot) permettrait aussi de supprimer les problèmes reliés à la fiabilité du matériel. De plus en plus courant de nos jours, ces appareils électroniques contiennent majoritairement la technologie nécessaire au bon fonctionnement de nos applications associées à une flexibilité logicielle accrue. Seule la partie d'amplification et de transmission d'énergie devrait être incluse dans un module extérieur.

L'utilisation de l'époxy à deux phases a permis de supprimer toute infiltration de liquide dans l'implant et a permis d'obtenir une isolation hermétique adéquate; cependant l'époxy solide obtenu conduit à une utilisation unique de l'implant sans possibilité de remplacement de la batterie ou d'éventuelles modifications. Cette contrainte ainsi que les nombreuses itérations vers la création d'un moule convenable pour notre système, démontre une fois de plus l'impératif de disposer d'un boîtier scellé en titane pour répondre aux besoins de robustesse et d'isolation nécessaire dans ces applications. Ce type de boîtiers biocompatibles associés aux performances réalisées par notre système paveront ainsi la voie vers une application sécuritaire et efficace chez l'humain.

BIBLIOGRAPHIE

[1] ACCORNERO, N., BINI, G., LENZI, G.L., AND MANFREDI, M. (1977). Selective activation of peripheral nerve fiber groups of different diameter by triangular shaped stimulus pulses. Journal Physiol., Vol. 273, 539-560.

[2] Anatomie du système urinaire [En ligne] http://www.bioweb.lu/Anatomie/Rein/Sys_urin.htm. (page consultée le 15 septembre 2002)

[3] ARABI, K. (1994). Conception d'un microstimulateur neuromusculaire destiné à la récupération des dysfonctions urinaires. Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

[4] ARABI, K., SAWAN, M. (1996). Implantable multiprogrammable microstimulator dedicated to bladder control. Medical and Biological Engineering and Computing, Vol. 34, 9-12.

[5] BOYER, S., SAWAN, M., ABDEL-GAWAD, M., ROBIN, S., ELHILALI, M. (2000). Implantable selective stimulator to improve bladder voiding: Design and chronic experiments in dogs. IEEE Transactions on rehabilitation engineering. Vol. 8, No. 4, 464-470.

[6] BRADLEY W.E. (1977). Experience with electronic stimulation of the micturition reflex function. In FT hambrecht, JB Reswick (eds): Functionnal electrical stimulation, New York and Basel: Marcel Dekker, Inc,119.

[7] BRADLEY, W.E., TIMM, G.W. AND CHOU, S.N. ,(1971). A decade of experience with electronic stimulation of the micturition reflex. Urol. Int., 26:283.

[8] BRINDLEY, G. S. (1977). An implant to empty the bladder or close the urethra. Journal of Neurology, Neurosurgery, and Psychiatry. 40, 358-369.

[9] BRINDLEY, G.S., AND CRAGGS, M.D. (1980). A technique for anodally blocking large nerve fibers through chronically implanted electrodes. J. Neurol. Neurosurg. Psychiatry, 43:1083.

[10] BRINDLEY, G.S., POLKEY, C.S AND RUSHTON, D.N. (1982). Sacral anterior root stimulators for bladder control in paraplegia. Paraplegia, 20:365.

[11] BRUMMER, S.B. AND TURNER, M.J. (1975). Electrical stimulation of the nervous system: The principal of safe charge injection with noble metal electrodes. Bioelectr. B. 2:13-25.

[12] BRUMMER, S.B. AND TURNER, M.J. (1977). Electrical stimulation with Pt electrodes: I-A method for determination of "real" electrodes areas. IEEE Trans. Biomed. Eng. 24:436-439.

[13] BUBACK, D. (2001). The use of neuromodulation for treatment of urinary incontinence. AORN Journal, Vol.73, 176-190.

[14] CRAMPON, M.A. (1999). Conception et réalisation d'électrodes neuronales dédiées à des stimulateurs électroniques implantables. Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

[15] CREASEY G., H. (1993). Electrical stimulation of sacral roots for micturition after spinal cord injury. Urologics Clinics of North America, vol. 20, No. 3, 505-515.

[16] DONFACK, C. (2000). Caractérisation de contacts électrodes-tissus pour les stimulateurs neuro-musculaires implantables. Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

[17] DYMOND, A.M. (1976). Characteristics of the metal-tissue interface of stimulation electrodes. IEEE Trans. Biomed.Eng. 23:274-280.

[18] FANG, Z-P. AND MORTIMER, J.T. (1991). Selective activation of small motor axons by quasitrapezoidal current pulses. IEEE Trans. Biomed. Eng., Vol. 38 168-174.

[19] FRIEDMAN H., NASHOLD B.S. JR, GRIMES J. (1977). Electrical Stimulation of the Conus Medullaris in the paraplegic - A five year review. In FT hambrecht, JB Reswick (eds): Functionnal electrical stimulation, New York and Basel: Marcel Dekker, Inc,173.

[20] FRIEDMAN, H., NASHOLD, B., S., SENECHAL, P. (1972). Spinal cord stimulation and bladder function in normal and paraplegic animals. Journal of Neurosurg.. Vol.36, 430-436.

[21] GEIRSSON, G., FALL, M. et SULLIVAN, L. (1995). Clinical and urodynamic effects of intravesical capsaicin treatment in patients with chronic traumatic spinal detrusor hyperreflexia. J. Urol., 154, 1825.

[22] GREATBATCH, W. et HOLMES, C. (1991). History of implantable devices. IEEE Eng. Med. Bio., 38-41.

[23] GROAT, W.C. (1995). Mechanisms underlying the recovery of lower urinary tract function following spinal cord injury. Paraplegia, 33, 493.

[24] GRUNEWALD, V., BHADRA, N., CREASEY, G.H. et MORTIMER, J.T. (1998). Functional conditions of micturition induced by selective sacral anterior root stimulation: Experimental results in a canine animal model. World. J. Urol., 16, 329-336.

[25] HABIB, H.N. (1967). Experience and recent contributions in sacral nerve stimulation for both human and animal. Br. J. Urolo., 39:73.

[26] HABLER, H.J., JANIG, W. et KOLTZENBURG, M.(1990). Activation of unmyelinated afferent fibers by mechanical stimuli and inflammation of the urinary bladder in the cat. J. Physiol., 425, 545.

[27] HALD, T., AGRAWAL, G. et KANTREWITZ, A. (1966). Studies in stimulation of the bladder and its motor nerves. Surgery, 60, 848.

[28] HALEEM, A., S., BOEHM, F., LEGATT, A., D., KANTROWITZ, A., STONE, B., MELMAN, A. (1993). Sacral root stimulation for controlled micturition : Prevention of detrusor-external sphincter dyssynergia by intraoperative identification and selective section of sacral nerve branches. The Journal of Urology. Vol. 149, 1607-1612.

[29] HALVERSTED, D.B. (1971). Electrical stimulation of the human bladder, 3 years later. J. Urol., 106, 673.

[30] HASSOUNA, M., LI, J.S., ELHILALI, M. (1994). Dogs as an animal model for neurostimulation. Neurology and Urodynamics, vol. 13, 159-167.

[31] HEINE,J.P., SCHMIDT, R.A. AND TANAGHO, E.A. (1977). Intraspinal sacral root stimulation for controlled micturition. Invest. Urolo., 15:78.

[32] L'appareil urinaire [En ligne]http://cri-cirs-wnts.univ-lyon1.fr/Polycopies/HistologieFonctionnelleOrganes/Urinaire.(page consultée le 15 septembre 2002)

[33] ISHIGOOKA, M., HASHIMOTO, T., SASAGAWA, I. IZUMIYA, K. AND NAKADA, T. (1994) Modulation of the urethral pressure by high-frequency block stimulus in dogs. Eur. Urol., 25:334.

[34] KAPLAN, A.C., CHANCELLOR, M.B. et BLAIVAIS, J.G. (1991). Bladder and sphincter behaviour in patients with spinal cord lesions. J. Urol., 146, 113.

[35] KHALAF, I.M., TOPPERCER, A. et ELHILALI, M.M. (1979). Urethral pressure responses to detrusor stretch. Invest. Urol., 17, 141.

[36] KO, W.H., LIANG, S.P. et FUNG, C.D.F. (1977). Design of radio-frequency powered coils for implant instruments. Med. Biol. Eng. Comput., 15, 634-640.

[37] KOLDEWIJN, E.L., RIJKHOFF, N.J., VAN KERREBROECK, PH.E.V., DEBREYNE, F.M.J. AND WIJKSTRA H. (1992) Selective sacral root stimulation for bladder control: Acute experiments in a animal model. J. Urol., 151:1674.

[38] LI, J.S., HASSOUNA, M., SAWAN, M., DUVAL, F., ELHILALI, M.M. (1995). Long-term effect of sphincteric fatigue during bladder neurostimulation. Vol. 153, 38-242.

[39] MADERBACHER, H.G., KOFLER, A., FISCHER, J., STRASSER, F., HARING, B. et SPANUDAKIS (1999). Early temporary and late permanent loss of electromicturition through an anterior sacral root stimulator (Brindley)- A 12 years experience. European Urology, XIV Congress of the European Association of urology, Abstract 60.

[40] McDONAGH, R.P., FORSTER, D.M. et THOMAS, D.G. (1990). Urinary incontinence in spinal cord injury patients following complete sacral posterior rhizotomy. Br. J. Urol., 66, 618-622.

[41] MEDTRONIC (1999). Interstim® therapy for urinary control. Medtronic Neurological Inc., Minneapolis.

[42] MORTIMER,J., T., KAUFMAN, D. (1980). Intramuscular electrical stimulation tissue damage. Annals of biomedical engineering, Vol.8, 235-244.

[43] NASHOLD B.S., FRIEDMAN H., GLEN J.H., GRIMES J.H.,BARRY, W.F., AVERY R. (1972). Electromicturition in paraplegia. Arch. Surg.; 104:195.

[44] NEUROCONTROL CORPORATION (1998). VOCARE® bladder system, implantable functional neuromuscular stimulator. Neurocontrol Corporation, Ohio.

[45] RIJKHOFF, N. J. M., WIJKSTRA, H., VAN KERREBROECK P. E. V., DEBRUYNE F. M. J. (1997). Selective detrusor activation by electrical sacral nerve root stimulation in spinal cord injury. The Journal of Urology, Vol. 157, 1504-1508.

[46] RIJKHOFF, N., J. M., KOLDEWIJN, E. L., VAN KERREBROECK P. E. V., DEBRUYNE F. M. J., WIJKSTRA, H. (1994) Acute animal studies on the use of an anodal block to reduce urethral resistance in sacral root stimulation. IEEE Transactions on Rehabilitation Engineering. Vol. 2, No. 2, 92-98.

[47] ROBIN, S. (1998). Réalisation et test d'un système implantable dédié à la stimulation neurale sélective. Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

[48] ROBIN, S., SAWAN, M., HARVEY, J-F., ABDEL-GAWAD, M., ABDEL-BAKY, T.M., ELHILALI, M. M. (1997). A new implantable microstimulator dedicated to selective stimulation of the bladder. Proceedings of the 19th Int. Conf. IEEE/EMBS, Chicago, Il., USA.

[49] SAUERWEIN, D.H., DOMURATH, B.K.H. et KUTZENBERGER, J.F. (1999). Experience with sacral deafferentation and implantation of an anterior root stimulator following in 294 spinal-cord injury patients. J. Urol., 161, 274.

[50] SAWAN, M., DUVAL, F., HASSOUNA, M., ELHILALI, M. (1994). A new transcutaneous fully-programmable neural stimulator. International Journal of Microcomputer Application, Vol 13, No.3, 142-147.

[51] SAWAN, M., DUVAL, F., HASSOUNA, M., LI, J-S., ELHILALI, M., LACHANCE, J., LECLAIR, M., POURMEHDI, S., MOUINE, J. (June 1992) Computerized Transcutaneous Control of a multichannel implantable urinary prosthesis. IEEE Transactions on Biomedical engineering, Vol. 39, No. 6, 600-609.

[52] SAWAN, M., HASSOUNA, M., LI, J-S., DUVAL, F., ELHILALI, M. (1996). Stimulator design and subsequent stimulation parameter optimization for controlling micturition and reducing urethral resistance. IEEE Transactions on Rehabilitation Engineering, Vol. 4, No. 1, 39-46.

[53] SCHMIDT, R.A., BRUSCHINI, H. AND TANAGHO,E.A. (1979). Urinary bladder and sphincter responses to stimulation of dorsal and ventral sacral roots. Invest. Urol, 16:300.

[54] SCHNEIDER, E. (2001). Conception et évaluation d'un système de stimulation électrique neurale dédié à la réhabilitation des fonctions vésicales. Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

[55] SCHNEIDER, E., SAWAN, M., BOYER, S., ABDELKARIM, A, ELHILALI, M.M. (2000). Sphincter contraction inhibition and detrusor hyperreflexia prevention using selective stimulation : chronic experiments in dogs. ***IFESS***, Denmark,.

[56] SCHUMACHER, S., BROSS, S., SCHEEPE, J.R., SIEF, C., JUNEMANN, K.P. et ALKEN, P. (1999). Extradural cold block for selective stimulation of the bladder: Development of a new technique. J. Urol., 161, 950-954.

[57] SCHURCH, B., RODIC, B. et JEANMOND, D. (1997). Posterior sacral rhizotomy and intradural anterior sacral root stimulation for treatment of the spastic bladder in spinal cord injury. J.Urol., 157, 610-614.

[58] SCHURCH, B., STOHRER, M., KRAMER, G., SCHMID, D.M., GAUL, G. et HAURI, D. (2000). Botulinum-A Toxin for Treating Detrusor Hyperreflexia in Spinal Cord Injured Patients: New Alternative to Anticholinergic Drugs? Preliminary Results. Journal of Urology, 164, 692-698.

[59] SCOTT, F.B., QUESADA, E.M., CARDUS, D. et LASKOURSKI, T. (1965). Electronic bladder stimulation, dog and human experiments. Invest. Urol., 3, 231.

[60] SHAKER, H. S., TU, L. M., ROBIN, S., ARABI, K., HASSOUNA, M., SAWAN, M., ELHILALI, M. M. (1998). Reduction of bladder outlet resistance by selective sacral root stimulation using high-frequency blockade in dogs : An acute study. The Journal of Urology, Vol. 160, 901-907.

[61] SMITH, B., PECKHAM, P.H. et KEITH, M.W. (1987). An externally powered multichannel implantable stimulator for versatile control of paralysed muscle. IEEE Transactions on Bio-Medical Engineering, 34, 499-508.

[62] SOLOMONOW, M. (1984). External control of the neuromuscular system. IEEE Trans. Biomed. Eng. BME, 31:752.

[63] SOLOMONOW, M., ELDRED, E., LYMAN, J. AND FOSTER, J. (1983) Control of muscle contractile force through indirect high-frequency stimulation. Am. J. Phys. Med., 62:71.

[64] STOHRER, M: (1990). Alteration of the urinary tract after spinal cord injury: diagnosis, prevention, and therapy of the sequalae. World J. Urol., 7, 205.

[65] SWEENEY J.D., MORTIMER, J.T. AND BODNER, D.R. (1989). Acute animal studies on electrically induced collision block of pudendal nerve motor activities. Neurourolog. Urodyn., 8:521.

[66] TALLALA, A., BLOOM, J., W., QUANG, N. (1987). FES for Bladder : Direct or indirect Means?. PACE, Vol. 10, 240-245.

[67] TANENBAUM, A.S. (1989). Computer networks. NJ, Prentice-Hall.

[68] THUROFF, J.W., SCHMIDT, R.A., BAZEED, M.A. et TANAGHO, E.A. (1983). Chronic stimulation of sacral roots in dogs. Eur. Urol., 9, 102-108.

[69] TIMM G.W., BRADLEY, W.E. (1969). Electrostimulation of the urinary detrusor to effect contraction and evacuation., Invest. Urol, 6:562.

[70] VALIQUETTE, L., TESSIER, J., SCHICK, E.. Physiologie des voies urinaires et pharmacologie de l'appareil urinaire. Querin, S., "Physiopathologie des maladies du rein et des voies urinaires", Chapitre 3.

[71] WALTER, J., S., SIDAROUS, R., ROBINSON, C., J., WHEELER, J., S., WURSTER, R.,D. (1992). Comparaison of direct bladder and sacral nerve stimulation in spinal cats. Journal of Rehabilitation Research and Development, Vol.29, No 2, 13-22.

[72] WALTER, J., S., WHEELER, J., S., ROBINSON, C., J., WURSTER, R.,D. (1993). Inhibiting the hyperreflexic bladder with electrical stimulation in a spinal animal model. Neurolourology and Urodynamics, 12, 241-253.

[73] WALTER, J., S., WHEELER, J., S., ROBINSON, C., KHAN, T., J., WURSTER, R.,D. (1989). Urethral responses to sacral stimulation in chronic spinal dog. The Journal of Physiology, Vol. 257, 284-291.

[74] WELD, K.J., GRANEY, M.J. et DMOCHOWSKI, R.R. (2000). Clinical significance of detrusor sphincter dyssynergia type in patients with post-traumatic spinal cord injury. Urology, 56, 565-568.

[75] WOO, M.Y. AND CAMPBELL,B. (1964). Asynchronous firing and block of peripheral nerve conduction by 20Kc alternative current. Bull. LA, Neuro. Soc., 29:87.

[76] YALLA, S.V., ROSSIER, A.B., et FAM, B.A. (1976). Dyssynergic vesicourethral responses during bladder rehabilitation. J. Urol., 115, 575.

[77] ZHANG, T. AND JIANG D. (1987). Selective Stimulation in a nerve trunk and its application in urology. Proc. of the 9th Ann. Int. Conf., IEEE-EMBS, Boston USA, p.1040.

ANNEXE A :
SCHEMAS ET CODE DU MICROSTIMULATEUR A STIMULATIONS SELECTIVE ET PERMANENTE

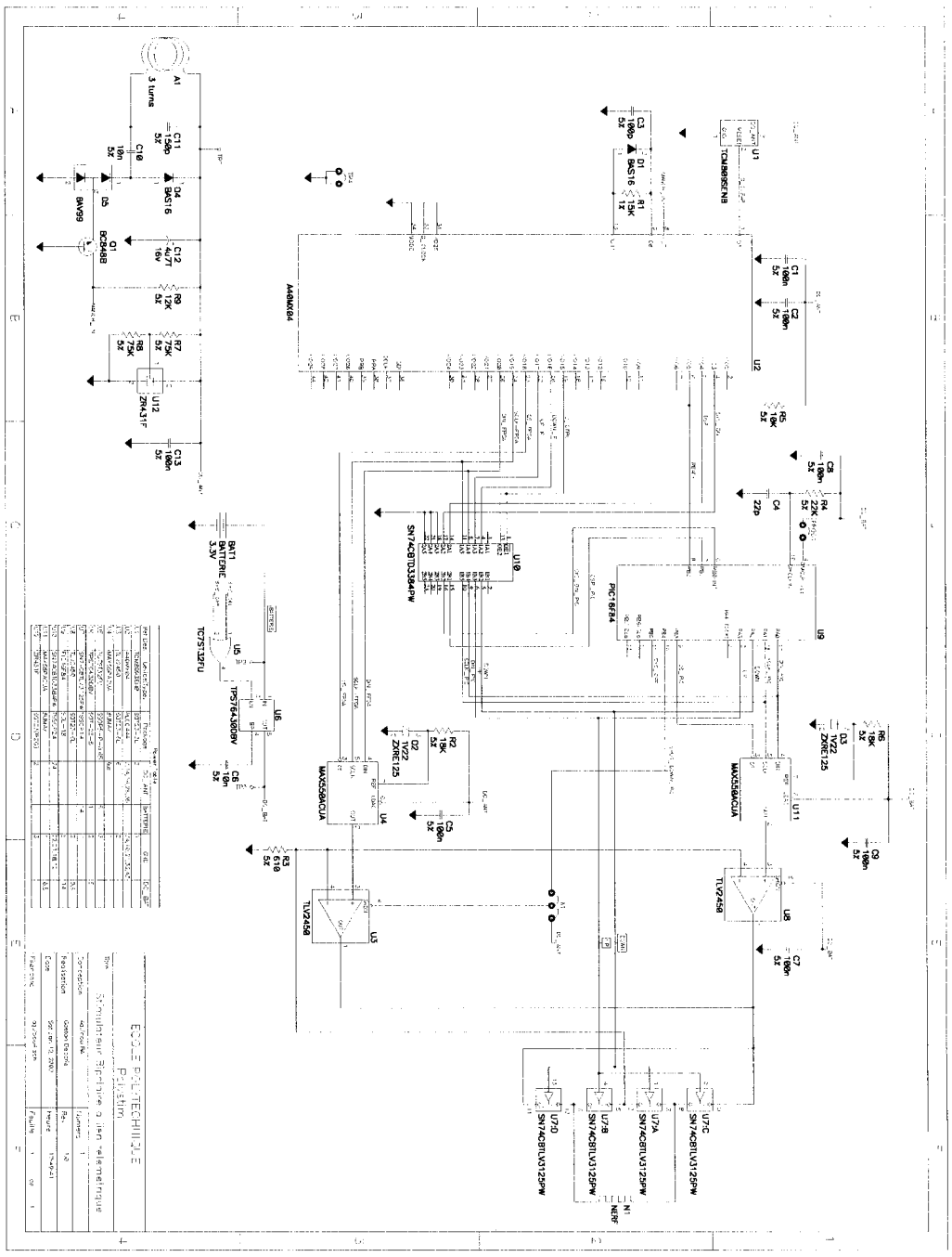


Figure A-1 : Schéma électrique du stimulateur sélectif et permanent

CAM350 V 5.0 : Sat Jan 12 17:47:13 2002 - (Untitled)

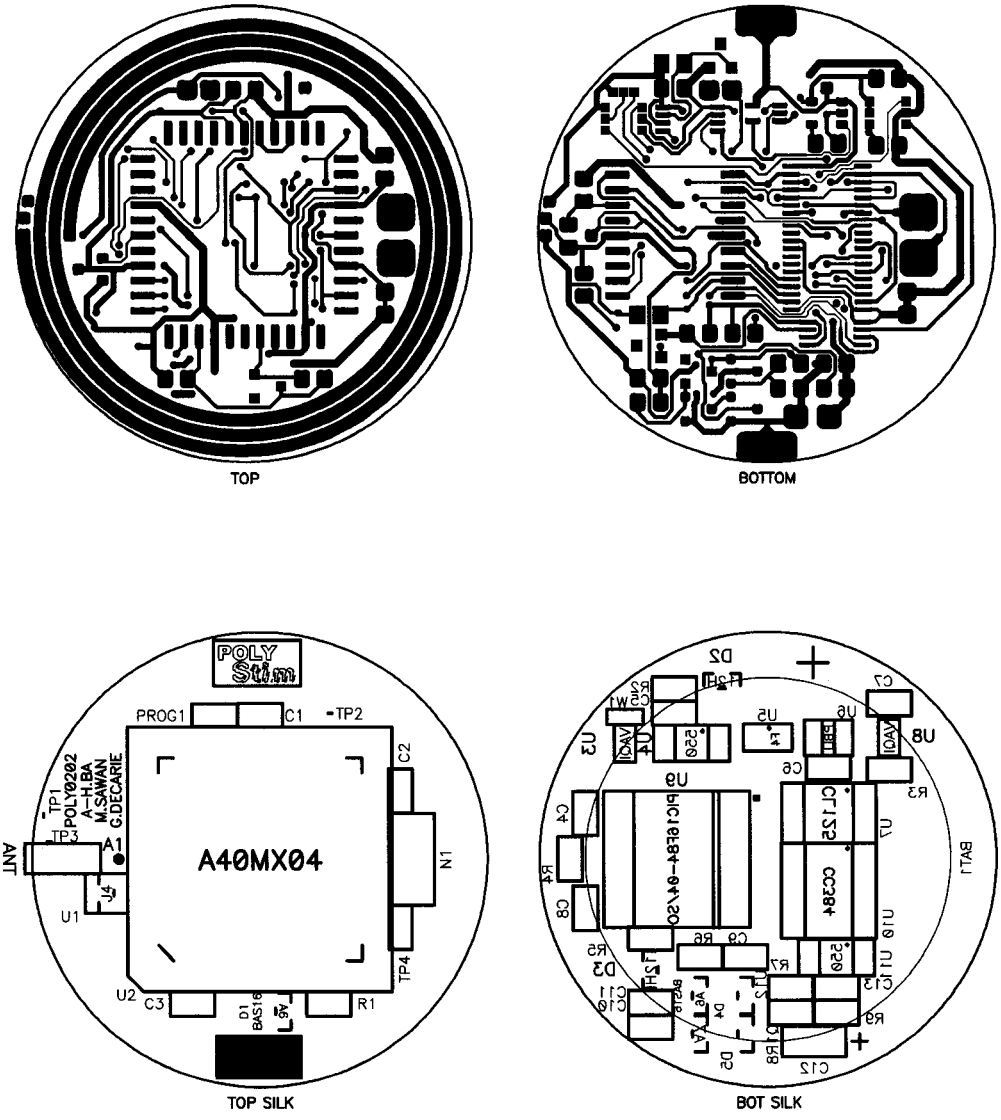


Figure A-2 : Dessin des masques à l'échelle 15/10 du stimulateur sélectif et permanent

```
;*****************************************************************
;  Filename:      My_Perm09.asm                                   *
;  Date:          20 Oct 2000                                     *
;  File Version:  1.2                                             *
;  Author:        E. Schneider                                    *
;  Company:       Polystim                                        *
;*****************************************************************
;  Modified by Aguibou BA Fevrier 2002                            *
;  Notes : program changed to allow PIC to work at startoff       *
;          and allows use of two different supply voltages        *
;*****************************************************************
;                                                                 *
;  Files required: P16F84.inc                                     *
;                                                                 *
;*****************************************************************
;                                                                 *
;  Notes: Program for the implant R4 with permanent stimulation   *
;                                                                 *
;*****************************************************************
;                          Default config                         *
;                  |     | BootUp |     |     |                  *
;         PinOut       | Stim  |/NoStim | Comm  | HighZ |        *
;                  | P  T |  P  T  |  P  T |  P  T |             *
;         RA0 = DIN     | '0' 0 | '0' 0  | 'Z' 1 | 'Z' 1 |       *
;         RA1 = SCLK    | '0' 0 | '0' 0  | '0' 0 | 'Z' 1 |       *
;         RA2 = DOWN    | '0' 0 | '0' 0  | 'Z' 1 | 'Z' 1 |       *
;         RA3 = UP      | '0' 0 | '0' 0  | 'Z' 1 | 'Z' 1 |       *
;(not used)  RA4 = SYS_OFF  | '0' 0 | '0' 0  | '0' 0 | '0' 0 |    *
;                  | *  * |  *  *  |  *  * |  *  * |             *
;         RB0 = SYS_ON  | 'Z' 1 | 'Z' 1  | 'Z' 1 | 'Z' 1 |      *
;         RB1 = DSR     | 'Z' 1 | 'Z' 1  | 'Z' 1 | 'Z' 1 |      *
;         RB2 = READY   | '0' 0 | '0' 0  | 'Z' 1 | 'Z' 1 |      *
;         RB3 = CS      | '1' 0 | '0' 0  | 'Z' 1 | 'Z' 1 |      *
;         RB4 = ANT_EN  | '1' 0 | '1' 0  | '1' 0 | '0' 0 |      *
;         RB5 = BAT_EN  | '1' 0 | '0' 0  | '0' 0 | '0' 0 |      *
;(not used)  RB6 = PROG2    | '0' 0 | '0' 0  | '0' 0 | '0' 0 |   *
;(not used)  RB7 = PROG3    | '0' 0 | '0' 0  | '0' 0 | '0' 0 |   *
;*****************************************************************
 list      p=16F84           ; list directive to define processor
 #include  <p16F84.inc>       ; processor specific variable definitions
 __CONFIG   _CP_OFF & _WDT_ON & _PWRTE_ON & _RC_OSC
;*****************************************************************
;                    MACRO DEFINITION                             *
;*****************************************************************
 #define   bank0  bcf STATUS,RP0
 #define   bank1  bsf STATUS,RP0

hi    macro port,pin
      bsf   port,pin
      endm
lo    macro port,pin
      bcf   port,pin
```

```
    endm
movlwf macro val,file
    movlw val
    movwf file
    endm
;*****************************************************************
;               CONSTANTE DEFINITION                              *
;*****************************************************************
RA          EQU    PORTA
RB          EQU    PORTB

DIN         EQU    D'0' ; (RA)
SCLK        EQU    D'1' ; (RA)
DOWN        EQU    D'2' ; (RA)
UP          EQU    D'3' ; (RA)
SYS_OFF     EQU    D'4' ; (RA)

SYS_ON      EQU    D'0' ; (RB)
DSR         EQU    D'1' ; (RB)
READY       EQU    D'2' ; (RB)
CS          EQU    D'3' ; (RB)
ANT_EN      EQU    D'4' ; (RB)
BAT_EN      EQU    D'5' ; (RB)
RB6         EQU    D'6' ; (RB)
RB7         EQU    D'7' ; (RB)

ComErr      EQU    D'0' ; (Flags)
CRCErr      EQU    D'1' ; (Flags)
Stdby       EQU    D'2' ; (Flags)
Shtdwn      EQU    D'3' ; (Flags)
NewData     EQU    D'4' ; (Flags)
SysOn       EQU    D'5' ; (Flags)

NbWordRead  EQU    D'6'

;*****************************************************************
;               VARIABLE DEFINITION                               *
;*****************************************************************
Temp1       EQU    0x0C ; for level 0 subroutines local use only
Temp2       EQU    0x0D ; for level 0 subroutines local use only
Temp3       EQU    0x0E ; for level 0 subroutines local use only
Temp4       EQU    0x0F ; for level 0 subroutines local use only

NbStim      EQU    0x10
Nb128Stim   EQU    0X11
Flags       EQU    0x12 ; b5=SysOn,b4=DataOK,b3=Shtdwn,b2=Stdby,b1=CRCErr,b0=ComErr
RAM         EQU    0x20 ; beginning of RAM space
CmdWord     EQU    0x20
CheckSum    EQU    0x25
CtrlDAC     EQU    0x20
Amplitude   EQU    0x21
PulseWide   EQU    0x22
```

```
Frequency    EQU    0x23
TimeOnOff    EQU    0x24
CtrlDACStdby EQU    0x25
AmplitudeZero EQU   0x26

;*********************************************************************
;              RESET VECTOR INITIALISATION                            *
;*********************************************************************
 ORG    0x000                   ; processor reset vector
                goto   Main             ; got to beginning of main program

;*********************************************************************
;              INTERRUPT REQUEST CODE                                 *
;*********************************************************************
 ORG    0x004                   ; interrupt vector location
                return                  ; return from interrupt

;*********************************************************************
;************************* MAIN PROGRAM CODE ************************
;*********************************************************************
Main
                clrf   RA               ; Init Port A
                movlwf B'00000000',RB   ; Init Port B
                bank1
                movlwf B'00001111',RA   ; Config Port A
                movlwf B'00000111',RB   ; Config Port B
                lo     OPTION_REG,T0CS
                bank0
                clrf   Flags
LoopMain
                clrwdt
                call   NoStim           ; Call "NoStim" if     SYS_ON=0 and DSR=0 and SysOn=0
                call   HighZ            ; Call "HighZ" if      SYS_ON=1 and DSR=0
                call   Communicate      ; Call "Communicate" if SYS_ON=1 and DSR=1
                call   Stimulate        ; Call "Stimulate" if  SYS_ON=0 and DSR=0 and SysOn=1
                goto   LoopMain

;*********************************************************************
;********************** END OF MAIN PROGRAM CODE **********************
;*********************************************************************

;*********************************************************************
;                  NOSTIM SUBROUTINE [LEVEL 1]                        *
; Action   : Force CPU in a wait loop while SYS_ON=0 and DSR=0        *
; Entries  : none                                                     *
; Subroutines : Standby                                               *
; Registers:                                                          *
;*********************************************************************
NoStim
                hi      RB,6
                btfsc   RB,SYS_ON
                goto    EndNoStim
```

```
                btfsc  RB,DSR
                goto   EndNoStim
                btfsc  Flags,SysOn
                goto   EndNoStim
                movlw  B'01000111'
                hi     INTCON,INTE
                call   Standby
                btfss  INTCON,INTF
                goto   NoStim
EndNoStim
                clrw
                call   Standby
                lo     RB,6
                return ; of NoStim subroutine

;*********************************************************************
;                HIGHZ SUBROUTINE [LEVEL 1]                          *
; Action  : Force CPU in a wait loop while SYS_ON=1 and DSR=0        *
;         Also connect DAC to RF power                               *
; Entries : none                                                     *
; Subroutines : Standby                                              *
; Registers:                                                         *
;*********************************************************************
HighZ
                btfss  RB,SYS_ON
                goto   EndEndHighZ
                btfsc  RB,DSR
                goto   EndEndHighZ
                clrw
                call   Standby
                btfss  RB,SYS_ON
                goto   EndEndHighZ
                btfsc  RB,DSR
                goto   EndEndHighZ
                bank1
                movlwf B'00001111',RA   ; Config Port A
                movlwf B'00001111',RB   ; Config Port B
                bank0
                movlwf B'00010000',RB   ; Init Port B *********
LoopHighZ
                btfss  RB,SYS_ON
                goto   EndHighZ
                btfsc  RB,DSR
                goto   EndHighZ
                movlw  B'00000111'
                hi     INTCON,INTE
                call   Standby
                btfss  INTCON,INTF
                goto   LoopHighZ
EndHighZ
                clrf   RA               ; Init Port A
                movlwf B'00000000',RB   ; Init Port B
```

```
                bank1
                movlwf  B'00001111',RA   ; Config Port A
                movlwf  B'00000111',RB   ; Config Port B
                bank0
EndEndHighZ
                return ; of HighZ subroutine

;*****************************************************************
;                COMMUNICATE SUBROUTINE [LEVEL 1]                *
; Action  : Establish communication with the FPGA to get new      *
;         stimulation parameters while SYS_ON=1 and DSR=1         *
; Entries : none                                                  *
; Subroutines : GetData,AnalyseData,Standby                       *
; Registers:                                                      *
;*****************************************************************
Communicate
                btfss   RB,SYS_ON
                goto    EndCommunicate
                btfss   RB,DSR
                goto    EndCommunicate
                clrw
                call    Standby
                btfss   RB,SYS_ON
                goto    EndCommunicate
                btfss   RB,DSR
                goto    EndCommunicate
                bank1
                movlwf  B'00001101',RA   ; Config Port A
                movlwf  B'00001111',RB   ; Config Port B
                bank0
                movlw   RAM
                call    GetData
                btfsc   Flags,ComErr
                goto    WaitEndCommunicate
                btfsc   Flags,CRCErr
                goto    WaitEndCommunicate
                movfw   CmdWord
                call    AnalyseData
WaitEndCommunicate
                btfss   RB,SYS_ON
                goto    EndCommunicate
                btfss   RB,DSR
                goto    EndCommunicate
                movlw   B'00000111'
                hi      INTCON,INTE
                call    Standby
                btfss   INTCON,INTF
                goto    WaitEndCommunicate
EndCommunicate
                clrf    RA              ; Init Port A
                movlwf  B'00000000',RB   ; Init Port B
                bank1
```

```
                movlwf B'00000111',RB   ; Config Port B
                movlwf B'00001111',RA   ; Config Port A
                bank0
                return ; of Communicate subroutine

;*****************************************************************
;               STIMULATE SUBROUTINE [LEVEL 1]                  *
; Action  : Force CPU stimulate while SYS_ON=0 and DSR=0         *
; Entries : none                                                 *
; Subroutines : SendData,UpDown,InterPhases,TimeOff,Standby      *
; Registers:                                                     *
;*****************************************************************
Stimulate

                clrf  RA            ; Init Port A
                movlwf B'00000000',RB   ; Init Port B
                bank1
                movlwf B'00000111',RB   ; Config Port B
                movlwf B'00000000',RA   ; Config Port A
                bank0
                btfsc  RB,SYS_ON
                goto   EndStimulate
                btfsc  RB,DSR
                goto   EndStimulate
                btfss  Flags,SysOn
                goto   EndStimulate
                clrw
                call   Standby
                btfsc  RB,SYS_ON
                goto   EndStimulate
                btfsc  RB,DSR
                goto   EndStimulate
                btfss  Flags,SysOn
                goto   EndStimulate
                movlwf B'11101001',CtrlDAC
                movlwf B'11111001',CtrlDACStdby
                clrf   AmplitudeZero
                lo     Amplitude,7      ; Security: to force I to be less than 1mA
                movlwf D'128',NbStim
                swapf  TimeOnOff,W
                andlw  B'00001111'
                movwf  Nb128Stim
LoopStimulate
                clrwdt
                movlwf B'00101000',RB        ; force BAT_EN=1,ANT_EN=1,CS=1,READY=0
                movlw  CtrlDAC
                call   SendData
                movfw  PulseWide
                call   UpDown
                movlw  CtrlDACStdby
                call   SendData
                movfw  Frequency
```

```
                call   InterPhases
                decf   NbStim,F
                bnz    ContinueStimulate
                movlwf D'128',NbStim
                decf   Nb128Stim,F
                bnz    ContinueStimulate
                movlwf B'00100000',RB          ;********** Sys_off
                movfw  TimeOnOff
                call   TimeOff
                swapf  TimeOnOff,W
                andlw  B'00001111'
                movwf  Nb128Stim
ContinueStimulate
                btfss  RB,SYS_ON
                goto   LoopStimulate
                movlwf B'00000000',RB
                clrf   Flags
EndStimulate
                clrf   RA              ; Init Port A
                movlwf B'00000000',RB    ; Init Port B
                bank1
                movlwf B'00000111',RB    ; Config Port B
                movlwf B'00001111',RA    ; Config Port A
                bank0
                return ; of Stimulate subroutine

;*********************************************************************
;                STANDBY SUBROUTINE [LOW LEVEL]                      *
; Action  : make the CPU go in standby for X secondes                 *
; Entries  : W[6]=Interrupt edge, W[2..0]=WDT prescaler bits          *
; Registers: OPTION_REG,INTCON                                        *
;*********************************************************************
Standby
                bank1
                iorlw  B'10111000'
                movwf  OPTION_REG
                lo     INTCON,INTF
                clrwdt
                nop
                sleep                  ; force CPU in standby
                nop
                lo     INTCON,INTE
                movlwf B'11011111',OPTION_REG  ; prog WDT for max delay
                bank0
EndStandby
                return ; of Standby subroutine

;*********************************************************************
;              GETDATA SUBROUTINE [LEVEL 0]                          *
; Action  : get parameters from FPGA via DIN port and write it to     *
;         RAM                                                         *
; Entries  : W = pointer in RAM                                       *
```

```
; Registers: Temp1,Temp2,Temp3,Temp4,Flags,FSR,INDF                    *
; Rq      : need to be in "Communicate" mode                           *
;***********************************************************************
GetData
                movwf  FSR          ; Initialise RAM pointer
                clrf   Flags
                movlwf NbWordRead,Temp3
                movlwf D'008',Temp2
                clrf   Temp4
LoopGetData
                btfss  RB,SYS_ON
                goto   ErrorComGetData
                btfss  RB,DSR       ; Test if data still available
                goto   ErrorComGetData
                rlf    Temp1,F
                btfss  RA,DIN       ; Temp1[0]=0 if DIN=0
                lo     Temp1,0
                btfsc  RA,DIN       ; Temp1[0]=1 if DIN=1
                hi     Temp1,0
                hi     RA,SCLK      ; Clock Registers in the FPGA
                lo     RA,SCLK
                decfsz Temp2,F
                goto   LoopGetData
                movf   Temp1,F
                bz     ErrorComGetData
                movfw  Temp1
                movwf  INDF         ; save Temp1 in RAM
                decf   Temp3,F
                bz     CheckSumGetData
                addwf  Temp4,F
                incf   FSR,F
                movlwf D'008',Temp2
                goto   LoopGetData
CheckSumGetData
                subwf  Temp4,W
                skpz
                hi     Flags,CRCErr
EndGetData
                return ; of GetData subroutine
ErrorComGetData
                hi     Flags,ComErr
                goto   EndGetData

;***********************************************************************
;               ANALYSEDATA SUBROUTINE [LEVEL 0]                       *
; Action   : Analyse data received from the FPGA                       *
; Entries  : W = commande word get from the FPGA                       *
; Registers: Temp1,Flags                                               *
;***********************************************************************
AnalyseData
                andlw  B'11111111'
                bz     EndAnalyseData
```

```
                movwf  Temp1
                incf   Temp1,W
                bz     EndAnalyseData
                swapf  Temp1,W
                xorwf  Temp1,W      ; Compare nibbles
                bnz    EndAnalyseData ; Finish if nibbles not identical
                btfss  Temp1,0
                goto   EndAnalyseData
                btfsc  Temp1,1
                hi     Flags,SysOn
EndAnalyseData
                return ; of AnalyseData subroutine
;*****************************************************************
;               SENDDATA SUBROUTINE [LEVEL 0]                    *
; Action  : send 2 words of RAM data pointed by W to DAC (DIN)    *
;         with control of SCLK and CS                             *
; Entries : W = pointer in RAM                                    *
; Registers: Temp1,Temp2,Temp3,FSR,INDF                           *
; Rq      : need to be in "Stimulate" mode                        *
;*****************************************************************
SendData
                movwf  FSR          ; initialise RAM pointer
                lo     RB,CS        ; force CS=0
                movlwf D'002',Temp2  ; send only 2 words per stream
Loop2SendData
                movlwf D'008',Temp1  ; 8 bits per word
                movfw  INDF
                movwf  Temp3
Loop1SendData
                btfss  Temp3,7
                lo     RA,DIN       ; DIN=0 if INDF[7]=0
                btfsc  Temp3,7
                hi     RA,DIN       ; DIN=1 if INDF[7]=1
                hi     RA,SCLK      ; force SCLK=1
                lo     RA,SCLK      ; force SCLK=0
                rlf    Temp3,F
                decfsz Temp1,F      ; test if end of word
                goto   Loop1SendData
                incf   FSR,F        ; increment RAM pointer
                decfsz Temp2,F      ; test if end of stream
                goto   Loop2SendData
                hi     RB,CS        ; force CS=1
                lo     RA,DIN       ; force DIN=0
EndSendData
                return ; of SendData subroutine
;*****************************************************************
;               UPDOWN SUBROUTINE [LEVEL 0]                      *
; Action  : ctrl signal UP & DOWN to commut currents in the nerve *
; Entries : W = time to be UP and DOWN                            *
; Registers: Temp1,Temp2                                          *
; Rq      : need to be in "Stimulate" mode                        *
;*****************************************************************
```

```
UpDown
                movwf  Temp1
                movwf  Temp2
                movlw  B'00001000'
                hi     RA,DOWN      ; force UP=0 and DOWN=1
Loop1UpDown
                decfsz Temp1,F
                goto   Loop1UpDown
                movwf  RA           ; force UP=1 and DOWN=0
Loop2UpDown
                decfsz Temp2,F
                goto   Loop2UpDown
                lo     RA,UP        ; force UP=0 and DOWN=0
EndUpDown
                return ; of UpDown subroutine

;*****************************************************************
;                INTERPHASES SUBROUTINE [LEVEL 0]                *
; Action  : make the CPU wait for X secondes depending on W value *
; Entries : W[7..6]=[PS1,PS0] of prescaler bits, W[5..4]=additional *
;         single sleep delay, W[3..0]=timer0 repeat time          *
; Registers: Temp1,Temp2,Temp3,INTCON,TMR0                        *
;*****************************************************************
InterPhases
                movwf  Temp1
                movwf  Temp2
                andlw  B'11000000'
                bz     AdditionalSleep
MainSleep
                clrc
                rrf    Temp1,F
                rrf    Temp1,F
                swapf  Temp1,F
                decf   Temp1,W
                iorlw  B'11111000'
                btfsc  RB,SYS_ON
                goto   EndInterPhases
                hi     INTCON,INTE
                call   Standby
                btfsc  INTCON,INTF
                goto   EndInterPhases
AdditionalSleep
                swapf  Temp2,W
                andlw  B'00000011'
                bz     EpsilonWait
                movwf  Temp3
LoopAddSleep
                btfsc  RB,SYS_ON
                goto   EndInterPhases
                movlw  B'11111000'
                hi     INTCON,INTE
                call   Standby
```

```
                btfsc   INTCON,INTF
                goto    EndInterPhases
                decfsz  Temp3,F
                goto    LoopAddSleep
EpsilonWait
                rlf     Temp2,F
                movlw   B'00011110'
                andwf   Temp2,F
                bz      EndInterPhases
Loop1EpsilonWait
                clrf    TMR0
                lo      INTCON,T0IF
Loop2EpsilonWait
                btfsc   RB,SYS_ON
                goto    EndInterPhases
                btfss   INTCON,T0IF
                goto    Loop2EpsilonWait
                decfsz  Temp2,F
                goto    Loop1EpsilonWait
EndInterPhases
                return ; of InterPhases subroutine

;*****************************************************************
;               TIMEOFF SUBROUTINE [LEVEL 0]                    *
; Action  : make the CPU go in standby for X secondes           *
; Entries : W[3..2]=repeated sleep, W[1..0]=[PS1,PS0] prescaler bits *
; Registers: Temp1,OPTION_REG,INTCON                            *
;*****************************************************************
TimeOff
                bank1
                movwf   Temp1
                iorlw   B'11111100'
                movwf   OPTION_REG
                bank0
                rrf     Temp1,F
                movlw   B'00000110'
                andwf   Temp1,F
                hi      INTCON,INTE
LoopTimeOff
                btfsc   RB,SYS_ON
                goto    ContinueTimeOff
                lo      INTCON,INTF
                clrwdt
                nop
                sleep           ; force CPU in standby
                nop
                btfsc   INTCON,INTF
                goto    ContinueTimeOff
                movf    Temp1,F
                bz      ContinueTimeOff
                decfsz  Temp1,F
                goto    LoopTimeOff
```

```
ContinueTimeOff
                bank1
                lo     INTCON,INTE
                movlwf  B'11011111',OPTION_REG ; prog WDT for max delay
                bank0
EndTimeOff
                return ; of TimeOff subroutine

 END
```

ANNEXE B :
CODES VHDL DU BLOC NUMERIQUE

```
-------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
-------------------------------------------------------------------------
-- File       : CANAL_Z.vhd
-- Author     : AGUIBOU BA
-- Company    : PolySTIM
-- Last update: 2002/09/24
-------------------------------------------------------------------------
-- Description: Ce module est un canal de stimulation. Apres avoir recu
-- les donnees du bloc DATA_RECOVERY, il detecte le type de stimulation
-- et l'active.
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use IEEE.numeric_std.all;

entity canal_z is
port (
      clk         :     in std_logic;
      rst_n       :     in std_logic;
      channel_in  :     in std_logic;
      receiving   :     in std_logic;
      mode_in     :     in std_logic_vector(1 downto 0);
      addr_in     :     in unsigned(4 downto 0);
      nb_pts_in   :     in std_logic_vector(3 downto 0);
      data_in     :     in std_logic_vector(7 downto 0);
      wr_n        :     in std_logic;
      tk_ctrl_in  :     in std_logic;
      tk_ctrl_out :     buffer std_logic;
      enab_gen    :     buffer std_logic;
      amp         :     out std_logic_vector(7 downto 0);
      test_cal    :     buffer std_logic;
      test_z      :     buffer std_logic;
      ps1         :     out std_logic;
      ps2         :     out std_logic;
      psr1        :     out std_logic;
      psr2        :     out std_logic;
      ns1         :     out std_logic;
      ns2         :     out std_logic;
      nt1         :     out std_logic;
      nt2         :     out std_logic;
      t1          :     out std_logic;
      pt1         :     out std_logic;
      pt2         :     out std_logic
      );
end canal_z;

architecture DATAFLOW of canal_z is
      component my_mux2a1
      generic (N : integer := 8);
      port (
            a, b        :     in std_logic_vector (N-1 downto 0);
```

```
      sel          :     in std_logic;
      q            :     buffer std_logic_vector (N-1 downto 0)
      );
end component;

component my_mux2a1_un
generic (N : integer := 8);
port (
      a, b         :     in unsigned (N-1 downto 0);
      sel          :     in std_logic;
      q            :     buffer unsigned (N-1 downto 0)
      );
end component;

component registre
generic (N : integer := 8);
port (
      clk          :     in STD_LOGIC;
      rst_n        :     in STD_LOGIC;
      enable       :     in STD_LOGIC;
      D            :     in STD_LOGIC_VECTOR ( N-1 downto 0);
      Q            :     buffer STD_LOGIC_vector ( N-1 downto 0)
      );
end component;

component RAM
generic     (
            addr_bits :integer:=5;

            data_bits :integer:=8
            );
port  (
      rst_n        :     in std_logic;
      clk          :     in std_logic;
      wr_n         :     in std_logic;
      rd_n         :     in std_logic;
      addr         :     in unsigned (addr_bits-1 downto 0);
      wr_data      :     in std_logic_vector(data_bits-1 downto 0);
      rd_data      :     buffer std_logic_vector(data_bits-1 downto
      0));
end component;

component msa_2
port  (
      clk          :     in std_logic;
      rst_n        :     in std_logic;
      receving     :     in std_logic;
      tk_ctrl      :     in std_logic;
      canal        :     in std_logic;
      fin_pulse    :     in std_logic;
      fin_width    :     in std_logic;
      mode         :     in std_logic_vector (1 downto 0);
      nb_pts       :     in std_logic_vector (3 downto 0);
```

```
      rd_n        :     buffer std_logic;
      RAZ         :     buffer std_logic;
      active      :     buffer std_logic;
      eval        :     buffer std_logic;
      rst_pulse_n :     buffer std_logic;
      rst_width_n :     buffer std_logic;
      en_gen_pt   :     buffer std_logic;
      en_swg      :     buffer std_logic;
      addr_ram    :     buffer unsigned (4 downto 0);
      sel_demux   :     buffer unsigned (2 downto 0)
      );
end component;

component GEN_PTS
port  (
      clk         :     in std_logic;
      rst_pulse   :     in std_logic;
      rst_width   :     in std_logic;
      enable      :     in std_logic;
      freq        :     in std_logic_vector(7 downto 0);
      pw          :     in std_logic_vector(7 downto 0);
      amp         :     in std_logic_vector(7 downto 0);
      signe_gen_pts:    buffer std_logic;
      amp_gen_pts :     buffer std_logic_vector(7 downto 0);
      fin_pulse   :     buffer std_logic;
      fin_width   :     buffer std_logic
      );
end component;

component demux
generic (N : integer := 8);
port  (
      rst_n       :     in std_logic;
      data        :     in std_logic_vector (N-1 downto 0);
      sel         :     in unsigned ( 2 downto 0) ;
      q0          :     buffer std_logic_vector (N-1 downto 0);
      q1          :     buffer std_logic_vector (N-1 downto 0);
      q2          :     buffer std_logic_vector (N-1 downto 0);
      q3          :     buffer std_logic_vector (N-1 downto 0);
      q4          :     buffer std_logic_vector (N-1 downto 0);
      q5          :     buffer std_logic_vector (N-1 downto 0);
      q6          :     buffer std_logic_vector (N-1 downto 0);
      q7          :     buffer std_logic_vector (N-1 downto 0)
      );
end component;

component SIG_TRANS
port  (
      test_cal    :     in  std_logic;
      test_z      :     in std_logic;
      stim        :     in std_logic;
      up_down     :     in std_logic;
      active      :     in std_logic;
```

```
      ps1          :     out std_logic;
      ps2          :     out std_logic;
      psr1         :     out std_logic;
      psr2         :     out std_logic;
      ns1          :     out std_logic;
      ns2          :     out std_logic;
      nt1          :     out std_logic;
      nt2          :     out std_logic;
      t1           :     out std_logic;
      pt1          :     out std_logic;
      pt2          :     out std_logic
      );
end component ;

component SWG_Z
port  (
      clk          :     in std_logic;
      rst_n        :     in std_logic;
      enable       :     in std_logic;
      mode         :     in std_logic;
      eval         :     in std_logic;
      lfp          :     in std_logic_vector(7 downto 0);
      hfp          :     in std_logic_vector(7 downto 0);
      lfw          :     in std_logic_vector(7 downto 0);
      hfw          :     in std_logic_vector(7 downto 0);
      lfa          :     in std_logic_vector(7 downto 0);
      hfa          :     in std_logic_vector(7 downto 0);
      pton         :     in std_logic_vector(3 downto 0);
      ptoff        :     in std_logic_vector(3 downto 0);
      ton_z        :     buffer std_logic;
      toff_z       :     buffer std_logic;
      signe_swg    :     buffer std_logic;
      amp_swg      :     buffer std_logic_vector(7 downto 0);
      active_swg   :     buffer std_logic
      );
end component;

component msa_z
port  (
      clk          :     in std_logic;
      rst_n        :     in std_logic;
      receving     :     in std_logic;
      active       :     in std_logic;
      eval         :     in std_logic;
      ton          :     in std_logic;
      toff         :     in std_logic;
      test_z       :     buffer std_logic;
      test_cal     :     buffer std_logic;
      stim         :     buffer std_logic
      );
end component ;
```

```
signal signe_pts,channel,
fin_pulse,fin_width,rst_pulse,rst_width,wr_n_in: std_logic;
signal stim,up_down, ton, toff ,eval: std_logic;
signal mode: std_logic_vector (1 downto 0);
signal nb_pts: std_logic_vector (3 downto 0);
signal rd_n ,RAZ ,active_pts ,en_gen_pt,en_swg :std_logic;
signal addr_ram:unsigned (4 downto 0);
signal sel_demux:unsigned (2 downto 0);
signal addr_mux : unsigned (5 downto 0);
signal addr_mux_in : unsigned (4 downto 0);
signal hfp,lfp,hfw,lfw,hfa,lfa: std_logic_vector(7 downto 0);
signal set_ton : std_logic_vector(3 downto 0);
signal set_toff : std_logic_vector(3 downto 0);
signal rd_data,amp_pts,X,ton_toff,in_reg,out_reg : std_logic_vector(7
downto 0);
signal addr_wr_in :unsigned (5 downto 0);
signal addr_wr_msa:unsigned (5 downto 0);
signal signe_swg ,signe,active: std_logic ;
signal amp_swg : std_logic_vector (7 downto 0);
signal amp_neg : std_logic_vector (7 downto 0);
signal active_swg : std_logic;
signal swg_mux : std_logic_vector (9 downto 0);
signal gen_pts_mux : std_logic_vector (9 downto 0);
signal out_mux : std_logic_vector (9 downto 0);

begin

      in_reg <= channel_in & tk_ctrl_in & mode_in & nb_pts_in;
      registre_0 : registre
      generic map (8)
      port map ( clk, RAZ, channel_in,in_reg,out_reg);
      channel     <= out_reg(7);
      tk_ctrl_out <= out_reg(6);
      mode        <= out_reg(5) & out_reg(4);
      nb_pts      <= out_reg(3) & out_reg(2)& out_reg(1) & out_reg(0);
      addr_wr_in  <= addr_in & wr_n;
      addr_wr_msa <= addr_ram & '1';
      my_mux2a1_un_0 : my_mux2a1_un
      generic map (6)
      port map ( addr_wr_in,addr_wr_msa, tk_ctrl_out, addr_mux);
      wr_n_in          <= addr_mux (0);
      addr_mux_in      <= addr_mux (5) & addr_mux (4) & addr_mux (3) &
      addr_mux (2) & addr_mux (1) ;
      GEN_PTS_0 : GEN_PTS
      port map
      (clk,rst_pulse,rst_width,en_gen_pt,lfp,lfw,lfa,signe_pts,amp_pts,f
      in_pulse,fin_width);
      msa_2_0 : msa_2
      port map
      (clk,rst_n,receiving,tk_ctrl_out,channel,fin_pulse,fin_width,mode,
      nb_pts,rd_n,RAZ,active_pts,eval,rst_pulse,rst_width,en_gen_pt,en_s
      wg,addr_ram,sel_demux );
      RAM_0 :RAM
```

```
      port map(RAZ , clk, wr_n_in, rd_n ,addr_mux_in, data_in,rd_data);
      demux_0 : demux
      port map ( RAZ,
      rd_data,sel_demux,lfp,hfp,lfw,hfw,lfa,hfa,ton_toff,X );
      set_ton     <= ton_toff(7) & ton_toff(6) & ton_toff(5) &
      ton_toff(4) ;
      set_toff    <= ton_toff(3) & ton_toff(2) & ton_toff(1) &
      ton_toff(0) ;

      SWG_Z_0 : SWG_Z
      port map (clk,
      RAZ,en_swg,mode(1),eval,lfp,hfp,lfw,hfw,lfa,hfa,set_ton,
      set_toff,ton, toff,signe_swg,amp_swg,active_swg);

      swg_mux           <= signe_swg  & amp_swg & active_swg;
      gen_pts_mux       <= signe_pts  & amp_pts & active_pts;

      my_mux2a1_0 : my_mux2a1
      generic map (10)
      port map ( swg_mux,gen_pts_mux, active_pts, out_mux);

      enab_gen          <= en_gen_pt or en_swg;
      signe             <= out_mux(9);
      amp_neg           <= out_mux(8) & out_mux(7) & out_mux(6) &
            out_mux(5) & out_mux(4) & out_mux(3) & out_mux(2) &
            out_mux(1)  ;
      amp               <= not amp_neg;
      active            <= out_mux(0);

      MSA_Z_0 : msa_z
      port map ( clk, rst_n, receiving, active, eval, ton,toff, test_z,
      test_cal, stim);

      SIG_TRANS_0 : SIG_TRANS
      port map ( test_cal,test_z,stim, signe,active,
      ps1,ps2,psr1,psr2,ns1,ns2,nt1,nt2,t1,pt1,pt2);

end DATAFLOW;

-------------------------------------------------------------------------
-- Title      : PFE
-------------------------------------------------------------------------
-- File       : CHECK_CRC.vhd
-- Author     : Dany-Sebastien Ly-Gagnon
-- Last update: 2002/04/17
-------------------------------------------------------------------------
-- Description: Verifie le CRC
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/04/17  1.0      venzz Created
-- 2002/08/00  2.0      GUIBS Verificateur de CRC du deuxieme Bloc
```

```
--------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;


entity check_crc is
    port (clk : in std_logic;
            clk2 : in std_logic;
        rst_n : in std_logic;
        enable : in std_logic;
        data : in std_logic;
        crc_ok : buffer std_logic);
end check_crc;

architecture DATAFLOW of check_crc is

    component shift_n
      generic (N : integer := 8);
      port (clk : in std_logic;
            rst_n : in std_logic;
            enable : in std_logic;
            din : in std_logic;
            dout : buffer std_logic_vector(N-1 downto 0));
    end component;

    component dff
      port (clk: in STD_LOGIC;
            rst_n: in STD_LOGIC;
            enable: in STD_LOGIC;
            D: in STD_LOGIC;
            Q: buffer STD_LOGIC;
            QN: buffer STD_LOGIC);
    end component;

    signal shift_in : std_logic;
    signal crc_vector : std_logic_vector(7 downto 0);
    signal crc_temp : std_logic;
    signal vdd : std_logic;

begin  -- DATAFLOW
    crc_reg : shift_n
      generic map (8)
      port map (clk, rst_n, enable, shift_in, crc_vector);

    dff_ccr : dff
      port map (crc_temp, rst_n ,enable, vdd, crc_ok, open);
            vdd <= '1';
    shift_in <= data xor crc_vector(7) ;
        crc_temp <=  not (crc_vector(0) or crc_vector(1) or
crc_vector(2) or crc_vector(3) or crc_vector(4) or crc_vector(5) or
crc_vector(6) or crc_vector(7));

end DATAFLOW;
```

```
--------------------------------------------------------------------------
-- Ecole Polytechnique de Montreal
-- Groupe de Recherche en Microelectronique
-- PolyStim - Equipe de Recherche en Neurotechnologies
-- Version originale: Eric Schneider
--  Nom du module: boundary
--  ce module génère les signaux de commande des grilles des transistors
--  des cicuits de permutation du courant dans les électrodes
-- Fichier: bound_scan.vhdl
--------------------------------------------------------------------------
--  Date Modification
--  98-11-24 Creation
--------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity reg is
      port( d : in std_logic;
            clk : in std_logic;
            clr : in std_logic;
            q : out std_logic);
end reg;

library ieee;
use ieee.std_logic_1164.all;
entity mux is
      port( in0 : in std_logic;
            in1 : in std_logic;
            sel : in std_logic;
            output : out std_logic);
end mux;


library ieee;
use ieee.std_logic_1164.all;
entity boundary is
      port( data        : in  std_logic_vector(7 downto 0);
            clr         : in  std_logic;
            test_clk    : in  std_logic;
            test_mode   : in  std_logic;
            sc_in       : in  std_logic;
            serial      : out std_logic);
end boundary;

library ieee;
use ieee.std_logic_1164.all;
Architecture behavior of reg is
begin
process(clk,clr)
begin
      if clr = '0' then q <= '0';
      elsif clk = '1' and clk'event then
```

```
            q <= d;
      end if;
end process;
end behavior;

library ieee;
use ieee.std_logic_1164.all;

Architecture behavior of mux is
begin
process(in0,in1,sel)
begin
      if sel = '0' then
            output <= in0;
      elsif sel = '1' then
            output <= in1;
      else output <= '-';
      end if;
end process;
end behavior;

library ieee;
use ieee.std_logic_1164.all;

Architecture structure of boundary is

component reg
      port( d : in std_logic;
            clk : in std_logic;
            clr : in std_logic;
            q : out std_logic);
end component;

component mux
      port( in0 : in std_logic;
            in1 : in std_logic;
            sel : in std_logic;
            output : out std_logic);
end component;

signal d,q : std_logic_vector(7 downto 0);

begin
      mux0 : mux port map (sc_in,data(0),test_mode,d(0));
      reg0 : reg port map (d(0),test_clk,clr,q(0));
      boucle : for i in 1 to 7 generate
            muxi : mux port map (q(i-1),data(i),test_mode,d(i));
            regi : reg port map (d(i),test_clk,clr,q(i));
      end generate boucle;
      serial <= q(7);

end structure;
```

```
--------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
--------------------------------------------------------------------------
-- File       : CNT_4.vhd
-- Author     : Aguibou BA
-- Last update: 2002/08/10
--------------------------------------------------------------------------
-- Description: Compteur de points 2 bits
--------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/08/10  1.0      GUIBS   Created
--------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity CNT_4 is
    port (clk : in std_logic;
          rst_n : in std_logic;
          EN : in std_logic;
          compte_4: buffer std_logic);
end CNT_4;

architecture DATAFLOW of CNT_4 is

    component cnt_n
      generic (N : integer := 8);
      port (clk : in std_logic;
            rst_n : in std_logic;
            enable : in std_logic;
            count : buffer std_logic_vector(N-1 downto 0));
    end component;

    component dff
      port (clk: in STD_LOGIC;
            rst_n: in STD_LOGIC;
            enable: in STD_LOGIC;
            D: in STD_LOGIC;
            Q: buffer STD_LOGIC;
            QN: buffer STD_LOGIC);
    end component;

      signal count :  std_logic_vector (2 downto 0);
      signal count_4 : std_logic;
      signal vdd : std_logic;

begin
      vdd<= '1';
    cnt_n_4: cnt_n generic map (3)
      port map (clk, rst_n, EN, count);
      count_4<= count (0) ;
```

```
   dff_cnt : dff
      port map (clk, rst_n,count_4, vdd, compte_4, open);

end DATAFLOW;

-------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
-------------------------------------------------------------------------
-- File       : CNT_8.vhd
-- Author     : Aguibou BA
-- Last update: 2002/08/10
-------------------------------------------------------------------------
-- Description: Compteur de points 3 bits
-------------------------------------------------------------------------
-- Date        Version  Author  Description
-- 2002/08/10  1.0      GUIBS   Created
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity CNT_8 is
    port (clk : in std_logic;
        rst_n : in std_logic;
        EN : in std_logic;
        compte_8: buffer std_logic);
end CNT_8;

architecture DATAFLOW of CNT_8 is

    component cnt_n
      generic (N : integer := 8);
      port (clk : in std_logic;
            rst_n : in std_logic;
            enable : in std_logic;
            count : buffer std_logic_vector(N-1 downto 0));
    end component;
      signal count :  std_logic_vector (2 downto 0);
begin

    cnt_n_8: cnt_n
      generic map (3)
      port map (clk, rst_n, EN, count);
      compte_8 <= count (2) and count (1) and count (0)  ;
end DATAFLOW;

-------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
-------------------------------------------------------------------------
-- File       : CNT_BITS.vhd
-- Author     : Aguibou BA
-- Last update: 2002/08/10
```

```
--------------------------------------------------------------------------
-- Description: Compteur de bits
--------------------------------------------------------------------------
-- Date        Version  Author  Description
-- 2002/08/10  1.0      GUIBS   Created
--------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity CNT_BITS is
    port (clk : in std_logic;
        rst_n : in std_logic;
        EN : in std_logic;
        perm, selec : buffer std_logic);
end CNT_BITS;

architecture DATAFLOW of CNT_BITS is

    component cnt_n
      generic (N : integer := 8);
      port (clk : in std_logic;
            rst_n : in std_logic;
            enable : in std_logic;
            count : buffer std_logic_vector(N-1 downto 0));
    end component;

    component dff
      port (clk: in STD_LOGIC;
            rst_n: in STD_LOGIC;
            enable: in STD_LOGIC;
            D: in STD_LOGIC;
            Q: buffer STD_LOGIC;
            QN: buffer STD_LOGIC);
    end component;

      signal count    :  std_logic_vector (7 downto 0);
      signal set_sel  :  std_logic;
      signal set_perm :  std_logic;
      signal vdd  :  std_logic;
begin

--------------------------------------------------------------------------
-- Bit counter
--------------------------------------------------------------------------
cnt_n_0 : cnt_n
      generic map (8)
      port map (clk, rst_n, EN, count);

    set_perm <= not count(7) and not count(6) and count(5) and
not count(4) and count(3) and not count(2) and not count(1) and
             not count(0);
```

```
    set_sel <= not count(7) and  not count(6) and count(5) and count(4)
and  count(3) and not count(2) and not count(1) and not count(0);
   vdd <= '1';

    dff_sel : dff
      port map (clk, rst_n,set_sel, vdd, selec, open);

    dff_perm: dff
      port map (clk, rst_n,set_perm, vdd, perm, open);

end DATAFLOW;

-------------------------------------------------------------------------
-- Title      : PFE
-------------------------------------------------------------------------
-- File       : CNT_N.vhd
-- Author     :   <venzz@darkstar.example.net>
-- Last update: 2002/04/17
-------------------------------------------------------------------------
-- Description: Compteur N bits
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/04/17  1.0      venzz Created
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_arith.all;
entity cnt_n is
    generic (N : integer := 8);
    port (clk : in std_logic;
                rst_n : in std_logic;
              enable : in std_logic;
              count : buffer std_logic_vector(N-1 downto 0));
end cnt_n;

architecture behavioral of cnt_n is
  signal COUNTaux : UNSIGNED(N-1 downto 0);
begin
  process (clk, rst_n)
  begin
    if (rst_n = '0') then
      COUNTaux <= CONV_UNSIGNED(0,N);
    elsif (clk'event and clk = '1') then
      if (enable = '1') then
        COUNTaux <= COUNTaux + 1;
      end if;
    end if;
  end process;
  count <= std_logic_vector(COUNTaux);

end behavioral;
```

```
-------------------------------------------------------------------------
-- Title      : PFE
-------------------------------------------------------------------------
-- File       : COMPARE_N.vhd
-- Author     :   <venzz@darkstar.example.net>
-- Last update: 2002/04/17
-------------------------------------------------------------------------
-- Description: Comparateur N bits
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/04/17  1.0      venzz Created
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity compare_n is

    generic (N : integer := 8);
    port (A : in std_logic_vector(N-1 downto 0);
        B : in std_logic_vector(N-1 downto 0);
        comp : buffer std_logic);
end compare_n;

architecture BEHAVIOR of compare_n is

    signal equal : std_logic_vector(N-1 downto 0);

begin
    process (A, B)
    begin
      if (A /= B) then
          comp <= '0';
      else
          comp <= '1';
      end if;
    end process;
end BEHAVIOR;

-------------------------------------------------------------------------
-- Title      : MS.C
-------------------------------------------------------------------------
-- File       : COMPTE_NB_PTS.vhd
-- Author     :   aguibou BA
-- Last update: 2002/08/10
-------------------------------------------------------------------------
-- Description: Detecte la fin d'un pattern de points
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/08/10  2.0      GUIBS Created
-------------------------------------------------------------------------
```

```
library ieee;
use ieee.std_logic_1164.all;

entity COMPTE_NB_PTS is
    port (clk : in std_logic;
        rst_n : in std_logic;
        valid : in std_logic;
        data : in std_logic;
            nb_pts : buffer std_logic_vector(3 downto 0));

end COMPTE_NB_PTS;

architecture structural of COMPTE_NB_PTS is

    component shift_n
      generic (N : integer);
      port (clk : in std_logic;
            rst_n : in std_logic;
            enable : in std_logic;
            din : in std_logic;
            dout : buffer std_logic_vector(N-1 downto 0));
    end component;


signal data_in : std_logic;
begin

    S1:     shift_n
      generic map (4)
      port map (clk, rst_n, valid, data, nb_pts);
end structural;

-------------------------------------------------------------------------
-- Title      : PFE
-------------------------------------------------------------------------
-- File       : CTRL_UNIT.vhd
-- Author     :   <venzz@darkstar.example.net>
-- Last update: 2002/04/17
-------------------------------------------------------------------------
-- Description: Unite de controle
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/04/17  1.0      venzz Created
-------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
entity ctrl_unit is

    port (clk : in std_logic;
        rst_n : in std_logic;
        hf_pulse : in std_logic;
```

```
        lf_pulse : in std_logic;
        busy : in std_logic;
        wsel : buffer std_logic;    -- 1 for high frequency, 0 for low
frequency
        start : buffer std_logic);
end ctrl_unit;

architecture BEHAVIOR of ctrl_unit is
    type STATES is (IDLE, STATE1A, STATE1B, STATE2A, STATE2B);
begin

    controller : process (clk, rst_n)
      variable state  : STATES;
    begin  -- process controller

      -- activities triggered by asynchronous reset (active low)
      if rst_n = '0' then
          state := IDLE;
          wsel <= '0';
          start <= '0';
      -- activities triggered by rising edge of clock
      elsif clk'event and clk = '1' then
          case state is

            ------------------------------------------------------------
            -- IDLE state
            ------------------------------------------------------------
            when IDLE =>
                if lf_pulse = '1' and busy /= '1' then
                  state := STATE1A;
                  wsel <= '0';
                  start <= '0';
                elsif hf_pulse = '1' and busy /= '1' then
                  state := STATE2A;
                  wsel <= '1';
                  start <= '0';
                else
                  state := IDLE;
                  wsel <= wsel;
                  start <= '1';
                end if;

            ------------------------------------------------------------
            -- STATE1 state: Low frequency Pulse
            ------------------------------------------------------------
            when STATE1A =>
                if busy = '1' then
                  state := STATE1A;
                  wsel <= wsel;
                  start <= '0';
                else
                  state := STATE1B;
                  wsel <= wsel;
```

```
                start <= '1';
            end if;

        when STATE1B =>
            state := IDLE;
            wsel <= wsel;
            start <= '1';

        ------------------------------------------------------------
        -- STATE2 state: High frequency Pulse
        ------------------------------------------------------------
        when STATE2A =>
            if busy = '1' then
              state := STATE2A;
              wsel <= wsel;
              start <= '0';
            else
              state := STATE2B;
              wsel <= wsel;
              start <= '1';
            end if;

        when STATE2B =>
            state := IDLE;
            wsel <= wsel;
            start <= '1';

      end case;

    end if;
  end process controller;

end BEHAVIOR;
--------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
--------------------------------------------------------------------------
-- File       : DATA_RECOVERY.vhd
-- Author     : Aguibou BA
-- Last update: 2002/08/10
--------------------------------------------------------------------------
-- Description: Ce module effectue la recuperation des
-- donnees sur la ligne serielle
--------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/08/10  1.0      GUIBS   Created
--------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity data_recovery is
```

```
    port (clk : in std_logic;
        rst_n : in std_logic;

        ------------------------------------------------------------------
        -- Input Data
        ------------------------------------------------------------------
        data : in std_logic;
        receiving : in std_logic;

        ------------------------------------------------------------------
        -- Control and activation output
        ------------------------------------------------------------------
        mode_out : buffer std_logic_vector(1 downto 0);
        channel  : buffer std_logic_vector(1 downto 0);
        addr     : buffer unsigned(4 downto 0);
        nb_pts   : buffer std_logic_vector(3 downto 0);
        data_out : buffer std_logic_vector(9 downto 0);
        wr_n     : buffer std_logic;
        valid    : buffer std_logic;
        crc_ok   : buffer std_logic;
        tk_ctrl  : buffer std_logic);

end data_recovery;

architecture DATAFLOW of data_recovery is

---------------------------------------------------------------------------
-- Components: header, six_one and shift_n
---------------------------------------------------------------------------
    component header_detector

    port (
            clk              : in std_logic;
            rst_n            : in std_logic;
            enable           : in std_logic;
            six_one          : in std_logic;
            data             : in std_logic;
            valid            : buffer std_logic;
            mode             : buffer std_logic;
            mode_2           : buffer std_logic;
            canal            : buffer std_logic_vector (1 downto 0);
            clr_reg_header_n: buffer std_logic);
   end component;

    component six_one
            port (
            clk         : in std_logic;
            rst_n       : in std_logic;
            enable      : in std_logic;
            data        : in std_logic;
            five_one    : buffer std_logic;
            six_one     : buffer std_logic);
    end component;
```

```
component check_crc
        port (
        clk          : in std_logic;
        clk2         : in std_logic;
        rst_n        : in std_logic;
        enable       : in std_logic;
        data         : in std_logic;
        crc_ok       : buffer std_logic);
end component;

component dff
  port (
        clk          : in STD_LOGIC;
        rst_n        : in STD_LOGIC;
        enable       : in STD_LOGIC;
        D            : in STD_LOGIC;
        Q            : buffer STD_LOGIC;
        QN           : buffer STD_LOGIC);
end component;

component CNT_4
port (
        clk          : in std_logic;
        rst_n        : in std_logic;
        EN           : in std_logic;
        compte_4     : buffer std_logic);
end component;

        component CNT_8
 port (
        clk          : in std_logic;
        rst_n        : in std_logic;
        EN           : in std_logic;
        compte_8     : buffer std_logic);
        end component;

        component CNT_BITS
 port (
        clk          : in std_logic;
        rst_n        : in std_logic;
        EN           : in std_logic;
        perm, selec  : buffer std_logic);
        end component;

        component SHIFT_9
port (
        clk          : in std_logic;
        rst_n        : in std_logic;
        enable       : in std_logic;
        data         : in std_logic;
        data_out     : buffer std_logic_vector (9 downto 0));
        end component;
```

```
        component msa_1
port (
        clk         : in std_logic;
        rst_n       : in std_logic;
        valid       : in std_logic;
        mode_in     : in std_logic_vector(1 downto 0);
        canal_in    : in std_logic_vector(1 downto 0);
        crc_ok      : in std_logic;
        cpte_perm   : in std_logic;
        cpte_sele   : in std_logic;
        cpte_pts    : in std_logic;
        receving    : in std_logic;
        cnt_8       : in std_logic;
        nb_pts      : in std_logic_vector(3 downto 0);
        load        : buffer std_logic;
        en_cpt_pt   : buffer std_logic;
        en_cnt_8    : buffer std_logic;
        wr_n        : buffer std_logic;
        tk_ctrl     : buffer std_logic;
        RAZ         : buffer std_logic;
        mode_out    : buffer std_logic_vector(1 downto 0);
        canal_out   : buffer std_logic_vector(1 downto 0);
        addr_ram    : buffer unsigned(4 downto 0));
        end component;

        component COMPTE_NB_PTS
port (
        clk         : in std_logic;
        rst_n       : in std_logic;
        valid       : in std_logic;
        data        : in std_logic;
        nb_pts      : buffer std_logic_vector(3 downto 0));
        end component;

signal valid_cmd    : std_logic;
signal load         : std_logic;      -- Enables shift registers
signal RAZ          : std_logic;      -- Clear
signal RST          : std_logic;      -- Clears everything
signal five_one_s   : std_logic;      -- Five one detected
signal six_one_s    : std_logic;      -- Six  one detected
signal en_cpte      : std_logic;
signal nb_pts_ct    : std_logic;
signal en_cpte2     : std_logic;
signal cpte_4       : std_logic;
signal cpte_8       : std_logic;
signal mode         : std_logic;
signal mode2        : std_logic;
signal mode_entre   : std_logic_vector(1 downto 0);
signal canal_in     : std_logic_vector(1 downto 0);
signal perm         : std_logic;
signal selec        : std_logic;
signal vdd          : std_logic;
```

```
    signal en_set1      : std_logic;
    signal en_set2      : std_logic;
    signal clr_reg_header_n   : std_logic;

begin

    vdd <= '1';
-------------------------------------------------------------------------
-- Six consecutive one detected
-------------------------------------------------------------------------
six_one_0 : six_one
      port map (clk, rst_n, receiving, data, five_one_s, six_one_s);
-------------------------------------------------------------------------
-- Header detector
--    => shft_en = '1' when header succesfully encountered
--    => clr_reg_header_n ='0' when command 0101 encountered after
header detected
-------------------------------------------------------------------------
header_detector_0 : header_detector
      port map (clk, RST, vdd, six_one_s, data, valid, mode, mode2,
canal_in, clr_reg_header_n);
        valid_cmd <= (not five_one_s) and valid;
    mode_entre <= mode2 & mode ;
-------------------------------------------------------------------------
-- Crc check
-------------------------------------------------------------------------
check_crc_0 : check_crc
      port map (clk,cpte_8, RST, valid_cmd, data, crc_ok);
-------------------------------------------------------------------------
-- Bit counter
-------------------------------------------------------------------------
       cnt_n_bits : cnt_bits
      port map (clk, RST, valid_cmd, perm, selec);
-------------------------------------------------------------------------
-- NB_PTS counter
-------------------------------------------------------------------------
      nb_pts_ct <= (valid_cmd xor cpte_4) and (not five_one_s) ;
    compte_nb_pts_0 : compte_nb_pts
      port map (clk, RST, nb_pts_ct, data, nb_pts);
-------------------------------------------------------------------------
-- SHIFT 9
-------------------------------------------------------------------------
shift_9_0 : shift_9
      port map (clk, RST, valid_cmd, data, data_out);
-------------------------------------------------------------------------
-- CNT8
-------------------------------------------------------------------------
      en_set1 <= en_cpte2 and (not five_one_s);
       cnt_8_0 : CNT_8
      port map (clk, RST, en_set1, cpte_8);
-------------------------------------------------------------------------
-- CNT4
```

```
------------------------------------------------------------------------
      en_set2 <= en_cpte and (not five_one_s);
    cnt_4_0 : CNT_4
      port map (clk, RST, en_set2, cpte_4);
------------------------------------------------------------------------
-- MSA_1
------------------------------------------------------------------------
      msa_1_0 : msa_1
      port map (clk,rst_n ,valid_cmd, mode_entre, canal_in, crc_ok,
perm, selec, cpte_4, receiving, pte_8,nb_pts,load,en_cpte,en_cpte2,wr_n,
                tk_ctrl, RAZ, mode_out, channel, addr);
RST <= RAZ and rst_n;
end DATAFLOW;

------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
------------------------------------------------------------------------
-- File       : DEC_MANCH.vhd
-- Author     : Aguibou BA
-- Last update: 2002/08/10
------------------------------------------------------------------------
-- Description: Decodeur manchester avec recuperation du signal
-- d'horloge
------------------------------------------------------------------------
-- Revisions  :
-- Date
-- 2002/04/17  1.0
------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;


entity dec_manch is

    port (
            manch_in    : in std_logic;
            rst_manch_n : in std_logic;
            data_out    : buffer std_logic;
            G_clock     : buffer std_logic;
            rst_n       : in std_logic);
end dec_manch;

architecture STRUCTURAL of dec_manch is

    component dff
      port (
            clk         : in STD_LOGIC;
            rst_n       : in STD_LOGIC;
            enable      : in STD_LOGIC;
            D           : in STD_LOGIC;
            Q           : buffer STD_LOGIC;
            QN          : buffer STD_LOGIC);
```

```
    end component;

signal vdd : std_logic;
signal Q1 : std_logic;
signal Q2 : std_logic;
signal manch_in_n : std_logic;
begin

      vdd         <= '1';
      manch_in_n  <= not manch_in;
      G_clock     <=  Q1 nor Q2;

      dff_1 : dff
      port map (manch_in, rst_manch_n, rst_n, vdd, Q1,open);

      dff_2 : dff
      port map (manch_in_n, rst_manch_n, rst_n, vdd, Q2,open);

      dff_3 : dff
      port map (G_clock, rst_n, vdd, manch_in, data_out,open);

end STRUCTURAL;

-------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
-------------------------------------------------------------------------
-- File       : DEMUX.vhd
-- Author     :   GUIBS
-- Last update: 2002/04/17
-------------------------------------------------------------------------
-- Description: Demultiplexeur
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/04/17  1.0      venzz Created
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity demux is
   generic (N : integer := 8);
       port (
            rst_n : in std_logic;
            data  : in std_logic_vector (N-1 downto 0);
            sel   : in unsigned ( 2 downto 0) ;
            q0    : buffer std_logic_vector (N-1 downto 0);
            q1    : buffer std_logic_vector (N-1 downto 0);
            q2    : buffer std_logic_vector (N-1 downto 0);
            q3    : buffer std_logic_vector (N-1 downto 0);
            q4    : buffer std_logic_vector (N-1 downto 0);
            q5    : buffer std_logic_vector (N-1 downto 0);
```

```
            q6    : buffer std_logic_vector (N-1 downto 0);
            q7    : buffer std_logic_vector (N-1 downto 0));
end demux;

architecture BEHAVIOR of demux is

begin
  process (rst_n , data, sel)

    begin

            if rst_n = '0' then
                  q0 <= "00000000";
                  q1 <= "00000000";
                  q2 <= "00000000";
                  q3 <= "00000000";
                  q4 <= "00000000";
                  q5 <= "00000000";
                  q6 <= "00000000";
                  q7 <= "00000000";

            else
                  case sel is
                        when "000" =>
                       q0 <= data;
                  when "001" =>
                        q1 <= data;
                  when "010" =>
                        q2 <= data;
                  when "011" =>
                        q3 <= data;
                  when "100" =>
                        q4 <= data;
                  when "101" =>
                        q5 <= data;
                  when "110" =>
                        q6 <= data;
                  when "111" =>
                        q7 <= data;
                  when others =>
                        null;
                   end case;
            end if;
   end process;
end BEHAVIOR;
-------------------------------------------------------------------------
-- DFF.vhd
-- Flip-Flop
-------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity dff is
```

```
    port (clk: in STD_LOGIC;
        rst_n: in STD_LOGIC;
        enable: in STD_LOGIC;
        D: in STD_LOGIC;
        Q: buffer STD_LOGIC;
        QN: buffer STD_LOGIC);
end dff;


architecture BEHAV of dff is
begin

    process (clk, rst_n)

    begin  -- process
      if rst_n = '0' then
          Q <= '0';
          QN <= '1';
      elsif clk'event and clk = '1' then
            if enable = '1' then
                Q <= D;
                QN <= not D;
            else
                  Q<= Q;
                  QN<=QN;
            end if;
      end if;
    end process;

end BEHAV;

-------------------------------------------------------------------------------
-- Title      : PFE
-------------------------------------------------------------------------------
-- File       : MUX2A1.vhd
-- Author     :   <venzz@darkstar.example.net>
-- Last update: 2002/04/17
-------------------------------------------------------------------------------
-- Description:
-------------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/04/17  1.0      venzz Created
-------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity dff_six1 is
      port (
            clk         : in  std_logic;
            rst_n       : in std_logic;
            six_one     : in std_logic;
            cnt         : in std_logic;
```

```
            Q           : buffer std_logic
            );
end dff_six1;

architecture struc of dff_six1 is

component mux2a1
    port (
            a, b  : in std_logic;
            sel   : in std_logic;
            q     : buffer std_logic);
end component;

    component dff
      port (
            clk   : in STD_LOGIC;
            rst_n : in STD_LOGIC;
            enable: in STD_LOGIC;
            D     : in STD_LOGIC;
            Q     : buffer STD_LOGIC;
            QN    : buffer STD_LOGIC);
    end component;

      signal a_mux, out_mux ,vdd: std_logic;

begin
vdd <= '1';
      multiplex : mux2a1
            port map(a_mux,vdd,six_one,out_mux);
      dff6 : dff
            port map(clk, rst_n, vdd, out_mux, Q,open);
A_mux <= not cnt and Q;
end struc;
--------------------------------------------------------------------------
-- FREQ_DIVIDER.vhd
-- Diviseur de frequence par N
--------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity freq_divider is

    generic (N : integer := 8);

    port (clk : in std_logic;
        rst_n : in std_logic;
        enable : in std_logic;
        clk_out : out std_logic);

end freq_divider;

architecture DATAFLOW of freq_divider is
```

```
  signal COUNTaux : UNSIGNED(N-1 downto 0);
begin

  process (clk, rst_n)
  begin

    if (rst_n = '0') then
      COUNTaux <= CONV_UNSIGNED(0,N);


    elsif (clk'event and clk = '1') then
      if (enable = '1') then
        COUNTaux <= COUNTaux - 1;
      end if;
    end if;

  end process;
  clk_out <= std_logic(COUNTaux(N-1));

end DATAFLOW;

library ieee;
use ieee.std_logic_1164.all;

entity freq_estim is
      port (
                  clk               : in std_logic;
                  rst_n             : in std_logic;
                  test_z            : in std_logic;
                  f_osc             : in std_logic;
                  test_clk          : in std_logic;
                  test_mode         : in std_logic;
                  sc_in             : in std_logic;
                  done              : out std_logic;
                  serial            : out std_logic
                  );
end freq_estim;

architecture struct of freq_estim is
      component eval_freq
      port(
                  data        : out std_logic_vector(7 downto 0);
                  done        : out std_logic;
                  test_z      : in std_logic;
                  por         : in std_logic;
                  clk         : in std_logic;
                  fosc        : in std_logic
            );
      end component ;

      component boundary
      port(
```

```
                data        : in  std_logic_vector(7 downto 0);
                clr         : in  std_logic;
                test_clk    : in  std_logic;
                test_mode   : in  std_logic;
                sc_in       : in  std_logic;
                serial      : out std_logic);
        end component;

        signal data : std_logic_vector ( 7 downto 0);

        begin
              ass1 : eval_freq
              port map ( data, done ,test_z, rst_n,clk, f_osc);

              ass2 : boundary
              port map ( data, rst_n, test_clk, test_mode, sc_in, serial);

end struct;
-------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
-------------------------------------------------------------------------
-- File       : Gen_pts.vhd
-- Author     : Guibs
-- Last update: 2002/04/17
-------------------------------------------------------------------------
-- Description: Ce module genere les points de stimulation en mode
-- stimulation Selective flexible.
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/04/17  1.0      venzz Created
-------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
entity GEN_PTS is

    port (
              clk               : in std_logic;
              rst_pulse         : in std_logic;
              rst_width         : in std_logic;
              enable            : in std_logic;
              freq              : in std_logic_vector(7 downto 0);
              pw                : in std_logic_vector(7 downto 0);
              amp               : in std_logic_vector(7 downto 0);
              signe_gen_pts     : buffer std_logic;
              amp_gen_pts       : buffer std_logic_vector(7 downto 0);
              fin_pulse         : buffer std_logic;
              fin_width         : buffer std_logic);

end GEN_PTS;
```

```
architecture DATAFLOW of GEN_PTS is
    component my_pgen
      generic (N : integer := 8);
      port (clk         : in std_logic;
            rst_n       : in std_logic;
            enable      : in std_logic;
            pfreq       : in std_logic_vector(N-1 downto 0);
            pulse       : buffer std_logic);
    end component;
    component freq_divider
      generic (N : integer := 8);
      port (clk         : in std_logic;
            rst_n       : in std_logic;
            enable      : in std_logic;
            clk_out     : out std_logic);

    end component;

    signal clk_4 : std_logic;

begin

      amp_gen_pts <= amp(7) & amp(6) & amp(5) & amp(4) & amp(3) & amp(2)
& amp(1) & '0';
      signe_gen_pts <= amp(0);

   ------------------------------------------------------------------------
    -- Frequency dividers
    ------------------------------------------------------------------------
    freq_divider_1 : freq_divider
      generic map (2)
      port map (clk, rst_pulse, enable, clk_4);
    ------------------------------------------------------------------------
    -- Pulse generator (High frequency: active for selective stimulation
only)
 ------------------------------------------------------------------------
       my_pgen_0 : my_pgen
           generic map (8)
           port map (clk_4, rst_pulse, enable,freq, fin_pulse);

------------------------------------------------------------------------
-- Pulse generator (Low frequency)
------------------------------------------------------------------------
       my_pgen_1 : my_pgen
           generic map (8)
           port map (clk, rst_width, enable,pw, fin_width);

end DATAFLOW;

------------------------------------------------------------------------
-- Title      : PFE
------------------------------------------------------------------------
-- File       : HEADER_DETECTOR.vhd
```

```
-- Author     :   <venzz@darkstar.example.net>
-- Last update: 2002/04/17
-------------------------------------------------------------------------
-- Description: Detecte et lit la commande envoyee
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/04/17  1.0      venzz Created
-- 2002/07/10           2.0   GUIBS Ajout de deux modes de
fonctionnement
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity header_detector is

    port (
                clk             : in std_logic;
                rst_n           :  in std_logic;
                enable          : in std_logic;
                six_one         : in std_logic;
                data            : in std_logic;
                valid           : buffer std_logic;
                mode            : buffer std_logic;
                mode_2          : buffer std_logic;
                canal           : buffer std_logic_vector (1 downto 0);
                clr_reg_header_n: buffer std_logic);

end header_detector;

architecture DATAFLOW of header_detector is

    component shift_n
      generic (N : integer := 8);
      port (clk         : in std_logic;
            rst_n       : in std_logic;
            enable      : in std_logic;
            din         : in std_logic;
            dout        : buffer std_logic_vector(N-1 downto 0));
    end component;

    component cnt_n
      generic (N : integer := 8);
      port (clk         : in std_logic;
            rst_n       : in std_logic;
            enable      : in std_logic;
            count       : buffer std_logic_vector(N-1 downto 0));
    end component;

    component dff
      port (clk         : in STD_LOGIC;
            rst_n       : in STD_LOGIC;
            enable      : in STD_LOGIC;
```

```
            D           : in STD_LOGIC;
            Q           : buffer STD_LOGIC;
            QN          : buffer STD_LOGIC);
    end component;

      component dff_six1
      port (
            clk         :     in std_logic;
            rst_n       :     in std_logic;
            six_one     :     in std_logic;
            cnt         :     in std_logic;
            Q           :     buffer std_logic
            );
      end component;

    signal shft10_en          : std_logic;
    signal shft10_en_temp     : std_logic;
    signal cnt10_en           : std_logic;
    signal cnt                : std_logic_vector(3 downto 0);
    signal cnt10              : std_logic;
    signal command            : std_logic_vector(9 downto 0);
    signal perm_set           : std_logic;
    signal perm               : std_logic;
    signal imped_set          : std_logic;
    signal imped              : std_logic;
    signal sel1_set           : std_logic;
    signal sel1               : std_logic;
    signal sel2_set           : std_logic;
    signal sel2               : std_logic;
    signal reset_set          : std_logic;
    signal vdd                : std_logic;
    signal cnt10_temp         : std_logic;

begin  -- Behavior

    vdd <= '1';
    ------------------------------------------------------------------------
    -- DFF input
    ------------------------------------------------------------------------

    dff_six_one : dff_six1
      port map (clk, rst_n, six_one, cnt10, shft10_en_temp);

      shft10_en <= shft10_en_temp and not cnt10;

    ------------------------------------------------------------------------
    -- 8 bit Counter
    ------------------------------------------------------------------------

    dff_cnt : dff
      port map (clk, rst_n, enable, shft10_en, cnt10_temp, open);
      cnt10_en <=  cnt10_temp and not cnt10;
    cnt_n_0 : cnt_n
```

```
      generic map (4)
      port map (clk, rst_n, cnt10_en, cnt);
    cnt10 <= cnt(3) and cnt(0) ;

  ------------------------------------------------------------------
  -- 8 bit Shift register
  ------------------------------------------------------------------
shft_n_0 : shift_n
generic map (10)
port map (clk, rst_n, shft10_en, data, command);

------------------------------------------------------------------ --
--State registers signals
------------------------------------------------------------------
perm_set <= cnt10 and command(7) and Not command(6) and command(5) and
            not command(4) and command(3) and not command(2) and
            command(1) and not command(0);

    imped_set <= cnt10 and command(7) and command(6) and not command(5)
and not command(4) and command(3) and command(2) and not command(1) and
            not command(0);

    sel1_set <= cnt10 and not command(7) and not command(6) and not
command(5) and not command(4) and command(3) and command(2) and
            command(1) and not command(0);


    sel2_set <= cnt10 and not command(7) and command(6) and command(5)
and command(4) and not command(3) and not command(2) and not command(1)
and not command(0);

    reset_set <= cnt10 and not command(7) and command(6) and not
command(5) and command(4) and not command(3) and command(2) and
            not command(1) and command(0);

    ------------------------------------------------------------------
-- State registers
    ------------------------------------------------------------------
dff_perm : dff Port map (clk, rst_n, enable, perm_set, perm, open);

    dff_impe : dff port map (clk , rst_n, enable, imped_set, imped,
open);

    dff_sel1 : dff port map (clk , rst_n, enable, sel1_set, sel1, open);

    dff_sel2 : dff port map (clk, rst_n, enable, sel2_set, sel2, open);

    dff_reset : dff Port map (clk, rst_n, enable, reset_set, open,
clr_reg_header_n);

    ------------------------------------------------------------------
-- Output logic
```

```
    ------------------------------------------------------------------
canal(1)<= command(9);
    canal(0)<= command(8);
    mode    <= perm or imped;
    mode_2  <= sel1 or imped ;
    valid   <= perm or sel1 or sel2 or imped;

end DATAFLOW;

--------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
--------------------------------------------------------------------------
-- File       : Ma_ram.vhd
-- Author     : Aguibou BA
-- Last update: 2002/08/10
--------------------------------------------------------------------------
-- Description: Bloc de RAM
--------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/08/10  1.0      GUIBS   Created
--------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity RAM is
  generic (addr_bits :integer;
           data_bits :integer);
  port (
            rst_n :in std_logic;
            clk :in std_logic;
            wr_n :in std_logic;
            rd_n :in std_logic;
            addr :in unsigned (addr_bits-1 downto 0);
            wr_data :in std_logic_vector(data_bits-1 downto 0);
            rd_data :buffer std_logic_vector(data_bits-1 downto 0)
        );
end RAM;

architecture rtl of RAM is

  constant low_address: natural := 0;
  constant high_address: natural := 31;

      type RAM_Image is array (2**addr_bits - 1 downto 0) of
std_logic_vector(data_bits - 1 downto 0);
      signal ma_ram     : RAM_Image;

begin
      process (rst_n, clk)
```

```
      variable adress : integer;
            begin
                  if rst_n = '0' then
              for add in low_address to high_address loop
                 ma_ram(add) <= "00000000";
                               rd_data   <= "00000000";
             end loop;
                  elsif clk'event and clk = '0' then
                        adress := to_integer (addr);

                        if wr_n <='0' then
                              ma_ram(adress) <= wr_data;
                              rd_data <= "00000000";
                        elsif rd_n <='0' then
                              rd_data <= ma_ram(adress);
                        else
                              rd_data <= rd_data;
                        end if;
                  end if;
      end process;
end;
-------------------------------------------------------------------------
-- Title      : PFE
-------------------------------------------------------------------------
-- File       : MODE_CTRL.vhd
-- Author     :   <venzz@darkstar.example.net>
-- Last update: 2002/04/17
-------------------------------------------------------------------------
-- Description: Determine le mode d'operation du CONTROLLER
-------------------------------------------------------------------------
-- Revisions  : 5 juin 2002 par Tommy Desilets
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity mode_ctrl is

    port (clk : in std_logic;
        rst_n : in std_logic;
        ctrl_en : in std_logic;
        mode : in std_logic;
        receiving : in std_logic;
        rst_all_n : buffer std_logic);

end mode_ctrl;

architecture BEHAVIOR of mode_ctrl is

    type STATES is (IDLE, PERM2, PERM1, SEL1, CLEAR_ALL);

begin  -- BEHAVIOR

    process (clk,rst_n)
```

```
      variable state  : STATES;
    begin
      if rst_n = '0' then
          state := CLEAR_ALL;
          rst_all_n <= '0';
      elsif clk='1' and clk'event then
          case state is
-------------------------------------------------------------------------
-- Idle state
-------------------------------------------------------------------------
            when IDLE =>
                if ctrl_en='1' and mode='0' then
                  state := PERM1;
                  rst_all_n <= '1';
                elsif ctrl_en='1' and mode='1' then
                  state := SEL1;
                  rst_all_n <= '1';
                else
                  state := IDLE;
                  rst_all_n <= '1';
                end if;

-------------------------------------------------------------------------
-- Permanent state
-------------------------------------------------------------------------
            when PERM1 =>
                if receiving='0' then
                  state := PERM2;
                  rst_all_n <= '1';
                else
                  state := PERM1;
                  rst_all_n <= '1';
                end if;
            when PERM2 =>
                if receiving='1' then
                  state := CLEAR_ALL;
                  rst_all_n <= '0';
                else
                  state := PERM2;
                  rst_all_n <= '1';
                end if;

-------------------------------------------------------------------------
-- Selective state
-------------------------------------------------------------------------
            when SEL1 =>
                if receiving='0' then
                  state := CLEAR_ALL;
                  rst_all_n <= '0';
                else
                  state := SEL1;
                  rst_all_n <= '1';
                end if;
```

```
-------------------------------------------------------------------------
-- Clear all state
-------------------------------------------------------------------------
            when CLEAR_ALL =>
                state := IDLE;
                rst_all_n <= '1';
          end case;
      end if;
    end process;

end BEHAVIOR;
-------------------------------------------------------------------------
-- Title      :   MAitrise
-------------------------------------------------------------------------
-- File       :   MSA_1.vhd
-- Author     :   Aguibou BA
-- Company    :   PolySTIM
-- Last update:   2002/07/08
-------------------------------------------------------------------------
-- Description:   Unite de controle superieure
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/07/08  1.0      Guibs      Created
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.all;

entity msa_1 is
    port (  clk              :     in std_logic;
            rst_n            :     in std_logic;
            valid            :     in std_logic;
            mode_in          :     in std_logic_vector(1 downto 0);
            canal_in         :     in std_logic_vector(1 downto 0);
            crc_ok           :     in std_logic;
            cpte_perm        :     in std_logic;
            cpte_sele        :     in std_logic;
            cpte_pts         :     in std_logic;
            receving         :     in std_logic;
            cnt_8            :     in std_logic;
            nb_pts           :     in std_logic_vector(3 downto 0);
            load             :     buffer std_logic;
            en_cpt_pt        :     buffer std_logic;
            en_cnt_8         :     buffer std_logic;
            wr_n             :     buffer std_logic;
            tk_ctrl          :     buffer std_logic;
            RAZ              :     buffer std_logic;
            mode_out         :     buffer std_logic_vector(1 downto 0);
            canal_out        :     buffer std_logic_vector(1 downto 0);
            addr_ram         :     buffer unsigned(4 downto 0));
end msa_1;
```

```
architecture BEHAVIOR of msa_1 is
    type STATES is IDLE, DETECT, PERM_PARAM, LOAD_P_P, SEL_PARAM,
LOAD_S_P, N_PTS, N_PTS1,PARAM_PTS,LOAD_PTS, STIM,WAIT_NEW);

      signal points          :     unsigned (3 downto 0);
      signal points2         :     std_logic_vector(3 downto 0);
      signal canal1          :     std_logic;
      signal canal2          :     std_logic;
      signal canal1_x        :     std_logic ;
      signal canal2_x        :     std_logic ;

begin

    controller : process
(clk,rst_n,points,points2,canal1,canal2,canal1_x,canal2_x,valid,mode_in,
canal_in,crc_ok,cpte_perm,receving,cnt_8,nb_pts,cpte_pts)

      variable state    :     STATES;
    begin
      points2     <= std_logic_vector(points - "0010") ;
      canal1_x    <= canal1;
      canal2_x    <= canal2;
      if rst_n = '0' then
            state             := IDLE;
            load              <='0';
            en_cpt_pt         <='0';
            en_cnt_8          <='0';
            wr_n              <='1';
            RAZ               <='1';
            tk_ctrl           <='0';
            mode_out          <="00";
            canal_out         <="00";
            addr_ram          <="00000";

      elsif clk'event and clk = '1' then
          case state is
      ------------------------------------------------------------------
      -- IDLE
      ------------------------------------------------------------------
            when IDLE =>
                  RAZ               <='0';
                  canal1            <= '0';
                  canal2            <= '0';
                  if receving = '1' then        -- Antenne RF active
                        state := DETECT;
                  else
                        state := IDLE;
                  end if;

      ------------------------------------------------------------------
      -- DETECT
      ------------------------------------------------------------------
```

```
when DETECT =>
      RAZ               <='1';
      if valid = '0' then -- Mot de commande valide detecte
            state       := DETECT;

      else
            case mode_in is
                  when "01"   =>    -- Mode Permanent
                        state       :=PERM_PARAM;
                        load        <='1';
                        en_cpt_pt   <='0';
                        en_cnt_8    <='1';
                        wr_n        <='1';
                        tk_ctrl     <='0';
                        mode_out    <= "01";

                        if canal_in = "01" then
                              canal_out   <= "01";
                              canal1      <= '1';
                        elsif canal_in = "10" then
                              canal_out   <= "10";
                              canal2      <= '1';
                        end if;
                        addr_ram    <="00000" ;

                  when "11"   =>    -- Mode impedance
                        state       := PERM_PARAM;
                        load        <='1';
                        en_cpt_pt   <='0';
                        en_cnt_8    <='1';
                        wr_n        <='1';
                        tk_ctrl     <='0';
                        mode_out    <= "11";

                        if canal_in = "01" then
                              canal_out   <= "01";
                              canal1      <= '1';
                        elsif canal_in = "10" then
                              canal_out   <= "10";
                              canal2      <= '1';
                        end if;
                        addr_ram          <="00000" ;

                  when "10" =>            -- Mode Selectif 1
                        state             := SEL_PARAM;
                        load              <='1';
                        en_cpt_pt         <='0';
                        en_cnt_8          <='1';
                        wr_n              <='1';
                        tk_ctrl           <='0';
                        mode_out          <="10";
                        if canal_in = "01" then
                              canal_out   <= "01";
```

```
                                    canal1      <= '1';
                                elsif canal_in = "10" then
                                    canal_out   <= "10";
                                    canal2      <= '1';
                                end if;
                                addr_ram        <="00000" ;

                            when "00" =>      -- Mode Selectif 2
                                state           := N_PTS;
                                load            <='0';
                                en_cpt_pt       <='1';
                                en_cnt_8        <='0';
                                wr_n            <='1';
                                tk_ctrl         <='0';
                                mode_out        <="00";
                                if canal_in = "01" then
                                    canal_out   <= "01";
                                    canal1      <= '1';
                                elsif canal_in = "10" then
                                    canal_out   <= "10";
                                    canal2      <= '1';
                                end if;
                                addr_ram        <="00000";

                            when others =>
                                null;

                        end case;
                    end if;
-------------------------------------------------------------------
-- PERM_PARAM
-------------------------------------------------------------------------
            when PERM_PARAM =>

                    if cnt_8 = '1' then     -- 8 bits dans le registre
                        state           := LOAD_P_P;
                        load            <='1';
                        en_cpt_pt       <='0';
                        en_cnt_8        <='1';
                        wr_n            <='0'; -- Active l'ecriture en
RAM
                        tk_ctrl         <='0';

                    else
                        state := PERM_PARAM;

                    end if;
-------------------------------------------------------------------------
-- LOAD_P_P
-------------------------------------------------------------------------
            when LOAD_P_P =>
                    if cpte_perm = '1' then  -- Ecriture complete en RAM
                        if crc_ok = '1' then        -- CRC Valide
```

```
                              state       := STIM ;
                              load        <='0';
                              en_cpt_pt   <='0';
                              en_cnt_8    <='0';
                              wr_n        <='1';
                              tk_ctrl     <='1'; -- control a La MSA du
canal
                        else                        -- Mauvais CRC
                              state       := WAIT_NEW ;
                              RAZ         <='0';
                              load        <='0';
                              en_cpt_pt   <='0';
                              en_cnt_8    <='0';
                              wr_n        <='1';
                              tk_ctrl     <='0';
                              mode_out    <="00";
                              canal_out   <="00";
                              addr_ram    <="00000";
                              if canal_in = "01" then
                                    canal1<= '0';
                              elsif canal_in = "10" then
                                    canal2<= '0';
                              end if;

                        end if ;
                  else        -- Autres parametres a ecrire
                        state := PERM_PARAM ;
                        wr_n              <='1';
                        addr_ram          <=addr_ram + "00001";
                  end if;
-------------------------------------------------------------------------
-- SEL_PARAM
-------------------------------------------------------------------------
            when SEL_PARAM =>
                  if cnt_8 = '1' then     -- 8 bits dans le registre
                        state             := LOAD_S_P;
                        load              <='1';
                        en_cpt_pt         <='0';
                        en_cnt_8          <='1';
                        wr_n              <='0'; -- Active l'ecriture en
RAM
                        tk_ctrl           <='0';
                  else
                        state             := SEL_PARAM;
                        wr_n              <='1';
                  end if;
-------------------------------------------------------------------------
-- LOAD_S_P
-------------------------------------------------------------------------
            when LOAD_S_P =>
                  if cpte_sele = '1' then       -- Ecriture complete en
RAM
                        if crc_ok = '1' then          -- CRC Valide
```

```
                              state       := STIM ;
                              load        <='0';
                              en_cpt_pt   <='0';
                              en_cnt_8    <='0';
                              wr_n        <='1';
                              tk_ctrl     <='1';      -- controle a La
MSA du canal
                                    else              -- Mauvais CRC
                              state := WAIT_NEW ;
                              RAZ         <='0';
                              load        <='0';
                              en_cpt_pt   <='0';
                              en_cnt_8    <='0';
                              wr_n        <='1';
                              tk_ctrl     <='0';
                              mode_out    <="00";
                              canal_out   <="00";
                              addr_ram    <="00000";
                              if canal_in = "01" then
                                    canal1<= '0';
                              elsif canal_in = "10" then
                                    canal2<= '0';
                              end if;
                        end if ;

                  else        -- Autres parametres a ecrire
                        state := SEL_PARAM ;
                        wr_n              <= '1';
                        addr_ram          <= addr_ram + "00001";
                  end if;

-------------------------------------------------------------------------
-- N_PTS
-------------------------------------------------------------------------
            when N_PTS =>
                  if cpte_pts = '1' then  -- Nombre de pts determine
                        state := N_PTS1 ;
                              load        <='1';
                              en_cpt_pt   <='0';
                              wr_n        <='1';
                              tk_ctrl     <='0';
                              points      <= "0000";

                        else
                              state       := N_PTS ;
                  end if;

-------------------------------------------------------------------------
-- N_PTS_1
-------------------------------------------------------------------------
            when N_PTS1 =>
                              state       := PARAM_PTS ;
                              en_cnt_8    <='1';
```

```
-------------------------------------------------------------------------
-- PARAM_PTS
-------------------------------------------------------------------------
            when PARAM_PTS =>
                  if cnt_8 = '1' then     -- 8 bits dans le registre
                        state             := LOAD_PTS;
                        load              <='1';
                        en_cpt_pt         <='0';
                        en_cnt_8          <='1';
                        wr_n        <='0'; -- Active l'ecriture en RAM
                        tk_ctrl           <='0';
                        points            <= points ;
                  else
                        state             := PARAM_PTS;
                  end if;

-------------------------------------------------------------------------
-- LOAD_PTS
-------------------------------------------------------------------------
            when LOAD_PTS =>
                  if ( points2  = nb_pts )then -- points complets en RAM
                        if cnt_8 = '1' then
                              if crc_ok = '1' then    -- CRC Valide
                                    state := STIM ;
                                    load        <='0';
                                    en_cpt_pt   <='0';
                                    en_cnt_8    <='0';
                                    wr_n        <='1';
                                    tk_ctrl   <='1';-- controle a La MSA
                                                    -- du canal
                              else                  -- Mauvais CRC
                                    state       := WAIT_NEW ;
                                    RAZ         <='0';
                                    load        <='0';
                                    en_cpt_pt   <='0';
                                    en_cnt_8    <='0';
                                    wr_n        <='1';
                                    tk_ctrl     <='0';
                                    mode_out    <="00";
                                    canal_out   <="00";
                                    addr_ram    <="00000";
                                    if canal_in = "01" then
                                          canal1<= '0';
                                    elsif canal_in = "10" then
                                          canal2<= '0';
                                    end if;
                              end if ;

                        else
                              state := LOAD_PTS;
                              load              <='0';
                              en_cpt_pt         <='0';
```

```
                        en_cnt_8          <='1';
                        wr_n              <='1';
                        tk_ctrl           <='0';
                  end if;
            else
                  state := PARAM_PTS ;
                  wr_n                    <='1';
                  addr_ram                <=addr_ram + "00001";
                  points                  <= points + "0001";
            end if;

-------------------------------------------------------------------------
-- STIM
-------------------------------------------------------------------------
      when STIM =>
            state                   := WAIT_NEW;
            RAZ                     <='0';
            load                    <='0';
            en_cpt_pt               <='0';
            en_cnt_8                <='0';
            wr_n                    <='1';
            mode_out                <="00";
            canal_out               <="00";
            addr_ram                <="00000";

-------------------------------------------------------------------------
-- WAIT_NEW
-------------------------------------------------------------------------
      when WAIT_NEW =>
            RAZ                     <='1';
            tk_ctrl                 <='0';
            if receving = '0' then
                  state             := IDLE;
            else
                  if valid = '0' then
                        state       := WAIT_NEW;
                  else
                        case canal_in is
                              when "01" =>
                              if canal1_x= '1' then
                                    state := STIM;
                              else
                                    case mode_in is
                                    when "01" => -- Mode Permanent
                                          state := PERM_PARAM;
                                                load        <='1';
                                                en_cpt_pt   <='0';
                                                en_cnt_8    <='1';
                                                wr_n        <='1';
                                                mode_out <= "01";
                                                canal_out<= "01";
                                                canal1      <= '1';
                                                addr_ram<=addr_ram;
```

```
                when "11"=>--Mode impedance
                      state :=  PERM_PARAM;
                            load        <='1';
                            en_cpt_pt   <='0';
                            en_cnt_8    <='1';
                            wr_n        <='1';
                            mode_out<= "11";
                            canal_out<= "01";
                            canal1 <= '1';
                      addr_ram <="00000" ;

                when "10" => --Mode Selectif 1
                            state := SEL_PARAM;
                            load        <='1';
                            en_cpt_pt   <='0';
                            en_cnt_8    <='1';
                            wr_n        <='1';
                            mode_out<="10";
                            canal_out   <= 01";
                            canal1      <= '1';
                           addr_ram<=addr_ram ;

                when "00" =>-- Mode Selectif 2
                            state := N_PTS;
                            load        <='0';
                            en_cpt_pt   <='1';
                            en_cnt_8    <='0';
                            wr_n        <='1';
                            mode_out    <="00";
                            canal_out   <= 01";
                            canal1      <= '1';
                      addr_ram    <="00000";

                      when others =>
                            null;
                      end case;
                end if;

when "10" =>
      if canal2_x= '1' then
                      state := STIM;
      else
                      case mode_in is
                      when "01" =>-- Mode Permanent
                            state := PERM_PARAM;
                            load        <='1';
                            en_cpt_pt   <='0';
                            en_cnt_8    <='1';
                            wr_n        <='1';
                            mode_out<= "01";
                            canal_out <= "10";
                            canal2      <= '1';
```

```
                                        addr_ram    <=addr_ram;

                                    when "11" =>-- Mode impedance
                                        state := PERM_PARAM;
                                              load        <='1';
                                              en_cpt_pt   <='0';
                                              en_cnt_8    <='1';
                                              wr_n        <='1';
                                              mode_out<= "11";
                                              canal_out<= "10";
                                              canal2      <= '1';
                                        addr_ram    <="00000" ;

                                    when "10" =>-- Mode Selectif 1
                                        state       := SEL_PARAM;
                                              load        <='1';
                                              en_cpt_pt   <='0';
                                              en_cnt_8    <='1';
                                              wr_n        <='1';
                                              mode_out    <="10";
                                              canal2      <= '1';
                                              canal_out<= "10";
                                              addr_ram<=addr_ram;

                                    when "00" =>--Mode Selectif 2
                                        state       := N_PTS;
                                              load        <='0';
                                              en_cpt_pt   <='1';
                                              en_cnt_8    <='0';
                                              wr_n        <='1';
                                              mode_out    <="00";
                                              canal_out<= "10";
                                              canal2      <= '1';
                                        addr_ram    <="00000";

                                        when others =>
                                              null;
                                        end case;
                                    end if;

                                when others =>
                                      null;
                                end case;
                            end if;
                        end if;
          when others =>
                null;
        end case;
    end if;
    end process controller;
    end BEHAVIOR;
    ---------------------------------------------------------------------
    -- Title      :   MAitrise
```

```
--------------------------------------------------------------------------
-- File       :   MSA_2.vhd
-- Author     :   Aguibou BA
-- Company    :   PolySTIM
-- Last update:   2002/07/26
--------------------------------------------------------------------------
-- Description:   Unite de controle de canal
--------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/07/26  1.0      Guibs       Created
--------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.all;

entity msa_2 is

    port (        clk               :     in std_logic;
                  rst_n             :     in std_logic;
                  receving          :     in std_logic;
                  tk_ctrl           :     in std_logic;
                  canal             :     in std_logic;
                  fin_pulse         :     in std_logic;
                  fin_width         :     in std_logic;
                  mode              :     in std_logic_vector(1 downto
0);
                  nb_pts            :     in std_logic_vector (3 downto
0);
                  rd_n              :     buffer std_logic;
                  RAZ               :     buffer std_logic;
                  active            :     buffer std_logic;
                  eval              :     buffer std_logic;
                  rst_pulse_n       :     buffer std_logic;
                  rst_width_n       :     buffer std_logic;
                  en_gen_pt         :     buffer std_logic;
                  en_swg            :     buffer std_logic;
                  addr_ram          :     buffer unsigned (4 downto 0);
                  sel_demux         :     buffer unsigned (2 downto 0)
            );
            end msa_2;


architecture BEHAVIOR of msa_2 is
    type STATES is (IDLE, WAIT_CTRL, RD_PARAM_PERM, STIM_PERM,
RD_PARAM_SEL,STIM_SEL, RD_FREQ_PTS, RD_PW_PTS, RD_PT, STIM_PTS,
WAIT_PULSE);

      signal points          :     unsigned (3 downto 0);
      signal points2         :     std_logic_vector(3 downto 0);
      signal param           :     unsigned (2 downto 0);
```

```
      signal param2           :     std_logic_vector (2 downto 0);

begin
      controller : process (clk, rst_n, receving, points,
points2,param,param2, mode, canal, tk_ctrl, fin_pulse, fin_width,
nb_pts)

      variable state    :STATES;

    begin
      points2     <=    std_logic_vector(points+ "0001") ;
      param2            <=    std_logic_vector(param) ;

      if rst_n = '0' then
            state       :=    IDLE;
            rd_n        <=    '1';
            RAZ         <=    '0';
            active      <=    '0';
            eval        <=    '0';
            en_gen_pt   <=    '0';


            en_swg      <=    '0';
            addr_ram    <=    "00000";
            sel_demux   <=    "000";
            rst_pulse_n <=    '0';
            rst_width_n <=    '0';
            param       <=    "000";

      -- Rising edge of clock
      elsif clk'event and clk = '1' then
          case state is

      ------------------------------------------------------------------
      -- IDLE
      ------------------------------------------------------------------
            when IDLE =>
                  RAZ                   <='1';
                  if canal = '1' then -- Antenne RF active
                        state := WAIT_CTRL;
                  else
                        state := IDLE;
                  end if;
      ------------------------------------------------------------------
      -- WAIT_CTRL
      ------------------------------------------------------------------
            when WAIT_CTRL =>
                  RAZ                    <='1';
                  if tk_ctrl = '0' then   -- Mot de commande valide
detecte
                        state := WAIT_CTRL;

                  else
```

```
                        case mode is
                              when "01"   =>    -- Mode Permanent
                                    state := RD_PARAM_PERM;
                                    rd_n        <='0';
                                    active      <='0';
                                    en_gen_pt   <='0';
                                    en_swg      <='0';
                                    addr_ram    <="00000";
                                    sel_demux   <="000";
                                    param       <="000";

                              when "11"   =>    -- Mode impedance
                                    state       := RD_PARAM_PERM;
                                    rd_n        <='0';
                                    eval        <='0';
                                    active      <='0';
                                    en_gen_pt   <='0';
                                    en_swg      <='0';
                                    addr_ram    <="00000";
                                    sel_demux   <="000";
                                    param       <="000";


                              when "10" =>      -- Mode Selectif 1
                                    state       := RD_PARAM_SEL;

                                    rd_n        <='0';
                                    active      <='0';
                                    en_gen_pt   <='0';
                                    en_swg      <='0';
                                    addr_ram    <="00000";
                                    sel_demux   <="000";
                                    param       <="000";

                              when "00" =>      -- Mode Selectif 2
                                    state       := RD_FREQ_PTS;

                                    rd_n        <='0';
                                    active      <='0';
                                    en_gen_pt   <='0';
                                    en_swg      <='0';
                                    addr_ram    <="00000";
                                    sel_demux   <="000";

                              when others =>
                                    null;

                        end case;
                  end if;

-------------------------------------------------------------------------
-- RD_PARAM_PERM
```

```
-----------------------------------------------------------------------
        when RD_PARAM_PERM =>

              if param2 = "011" then
                    if mode = "11" then
                          if receving ='1' then
                                state       := STIM_PERM;
                                rd_n        <='1';
                                RAZ         <='1';
                                active      <='0';
                                en_gen_pt   <='0';
                                eval        <='1';
                                en_swg      <='1';

                          else
                                state       := IDLE;
                                rd_n        <='1';
                                eval        <='0';
                                RAZ         <='0';
                                active      <='0';
                                en_gen_pt   <='0';
                                en_swg      <='0';
                                addr_ram    <="00000";
                                sel_demux   <="000";
                                param       <="000";
                          end if;

                    else
                          if receving = '0' then
                                state       := STIM_PERM;
                                rd_n        <='1';
                                RAZ         <='1';
                                active      <='0';
                                en_gen_pt   <='0';
                                en_swg      <='1';

                          else
                                state       := RD_PARAM_PERM;
                                rd_n        <='1';
                                RAZ         <='1';
                                active      <='0';
                                en_gen_pt   <='0';
                                en_swg      <='0';
                          end if;
                    end if;

              else
                    state                   := RD_PARAM_PERM;
                    addr_ram                <= addr_ram + "00001";
                    sel_demux               <= sel_demux + "010";
                    param                   <= param + "001";

              end if;
```

```
-------------------------------------------------------------------------
-- STIM_PERM
-------------------------------------------------------------------------
            when STIM_PERM =>
                  if mode = "11" then
                        if receving ='1' then
                              state             := STIM_PERM;
                        else
                              state             := IDLE;
                              rd_n              <='1';
                              RAZ               <='0';
                              active            <='0';
                              eval              <='0';
                              en_gen_pt         <='0';
                              en_swg            <='0';
                              addr_ram          <="00000";
                              sel_demux         <="000";
                              param             <="000";
                        end if;

                  else
                        if receving = '0' then
                              state             := STIM_PERM;

                        else
                              state             := IDLE;
                              rd_n              <='1';
                              RAZ               <='0';
                              active            <='0';
                              en_gen_pt         <='0';
                              en_swg            <='0';
                              addr_ram          <="00000";
                              sel_demux         <="000";
                              param             <="000";
                        end if;
                  end if;
-------------------------------------------------------------------------
-- RD_PARAM_SEL
-------------------------------------------------------------------------
            when RD_PARAM_SEL =>
                  if receving = '1' then
                        if param2 /= "110" then

                              state             := RD_PARAM_SEL;
                              addr_ram          <= addr_ram + "00001";
                              sel_demux         <= sel_demux + "001";
                              param             <= param + "001";

                        else
                              if receving = '1' then
                                    state       := STIM_SEL;
                                    rd_n        <='1';
                                    RAZ         <='1';
```

```
                                    active      <='0';
                                    en_gen_pt   <='0';
                                    en_swg      <='1';
                                    addr_ram    <="00000";

                              else
                                    state       := IDLE;
                                    rd_n        <='1';
                                    RAZ         <='0';
                                    active      <='0';
                                    en_gen_pt   <='0';
                                    en_swg      <='0';
                                    addr_ram    <="00000";
                                    sel_demux   <="000";
                                    param       <="000";

                              end if;
                        end if;

                  else
                        state := IDLE;
                        rd_n                    <='1';
                        RAZ                     <='0';
                        active                  <='0';
                        en_gen_pt               <='0';
                        en_swg                  <='0';
                        addr_ram                <="00000";
                        sel_demux               <="000";
                        param                   <="000";
                  end if;
-------------------------------------------------------------------------
-- STIM_SEL
-------------------------------------------------------------------------
            when STIM_SEL =>
                  if receving = '1' then
                        state                   := STIM_SEL;

                  else
                        state                   := IDLE;
                        rd_n                    <='1';
                        RAZ                     <='0';
                        active                  <='0';
                        en_gen_pt               <='0';
                        en_swg                  <='0';
                        addr_ram                <="00000";
                        sel_demux               <="000";
                        param                   <="000";
                  end if;

-------------------------------------------------------------------------
-- RD_FREQ_PTS
-------------------------------------------------------------------------
            when RD_FREQ_PTS =>
```

```
                  if receving ='1' then
                        state := RD_PW_PTS;
                        rd_n                    <='0';
                        RAZ                     <='1';
                        active                  <='0';
                        en_gen_pt               <='0';
                        en_swg                  <='0';
                        addr_ram                <="00001";
                        sel_demux               <="010";

                  else
                        state                   :=IDLE;

                        rd_n                    <='1';
                        RAZ                     <='0';
                        active                  <='0';
                        en_gen_pt               <='0';
                        en_swg                  <='0';
                        addr_ram                <="00000";
                        sel_demux               <="000";

                  end if;
-------------------------------------------------------------------------
-- RD_PW_PTS
-------------------------------------------------------------------------
            when RD_PW_PTS =>
                  if receving ='1' then
                        state                   := RD_PT;
                        rd_n                    <='0';
                        RAZ                     <='1';
                        active                  <='0';
                        en_gen_pt               <='0';
                        en_swg                  <='0';
                        addr_ram                <="00010";
                        sel_demux               <="100";
                        points                  <="0000";

                  else
                        state                   :=IDLE;
                        rd_n                    <='1';
                        RAZ                     <='0';
                        active                  <='0';
                        en_gen_pt               <='0';
                        en_swg                  <='0';
                        addr_ram                <="00000";
                        sel_demux               <="000";

                  end if;
-------------------------------------------------------------------------
--  RD_PT
-------------------------------------------------------------------------
            when RD_PT =>
                  if receving ='1' then
```

```
                        state                   := STIM_PTS;
                        rd_n                    <='1';
                        RAZ                     <='1';
                        rst_pulse_n             <='1';
                        rst_width_n             <='1';
                        active                  <='1';
                        en_gen_pt               <='1';
                        en_swg                  <='0';
                        addr_ram                <=addr_ram;
                        sel_demux               <=sel_demux;

                  else
                        state                   :=IDLE;
                        rst_pulse_n             <='0';
                        rst_width_n             <='0';
                        rd_n                    <='1';
                        RAZ                     <='0';
                        active                  <='0';
                        en_gen_pt               <='0';
                        en_swg                  <='0';
                        addr_ram                <="00000";
                        sel_demux               <="000";

                  end if;
-------------------------------------------------------------------------
-- STIM_PTS
-------------------------------------------------------------------------
      when STIM_PTS =>
                  if receving ='1' then
                        if fin_width ='0' then
                              state             := STIM_PTS;

                        elsif nb_pts = points2 then
                              state             := WAIT_PULSE;
                              rst_width_n       <='0';
                              active            <='0';
                              en_gen_pt         <='1';
                              addr_ram          <=addr_ram;
                              sel_demux         <=sel_demux;

                        else
                              state             := RD_PT;
                              rst_width_n       <='0';
                              rd_n              <='0';

                              addr_ram          <=addr_ram + "00001";
                              sel_demux         <=sel_demux ;
                              points            <= points + "0001" ;

                        end if;

                  else
                        state                   :=IDLE;
```

```
                        rst_pulse_n             <='0';
                        rst_width_n             <='0';
                        rd_n                    <='1';
                        RAZ                     <='0';
                        active                  <='0';
                        en_gen_pt               <='0';
                        en_swg                  <='0';
                        addr_ram                <="00000";
                        sel_demux               <="000";

                  end if;
-------------------------------------------------------------------------
-- WAIT_PULSE
-------------------------------------------------------------------------
            when WAIT_PULSE =>
                  if receving ='1' then
                        if fin_pulse ='0' then
                              state             := WAIT_PULSE;

                        else
                              state             := RD_PT;
                              rst_pulse_n       <='0';
                              rd_n              <='0';
                              RAZ               <='1';
                              active            <='1';
                              en_gen_pt         <='1';
                              en_swg            <='0';
                              addr_ram          <="00010";
                              sel_demux         <="100";
                              points            <="0000";

                        end if;

                  else
                        state                   :=IDLE;

                        rst_pulse_n             <='0';
                        rst_width_n             <='0';
                        rd_n                    <='1';
                        RAZ                     <='0';
                        active                  <='0';
                        en_gen_pt               <='0';
                        en_swg                  <='0';
                        addr_ram                <="00000";
                        sel_demux               <="000";

                  end if;

      when others =>
            null;
    end case;
end if;
```

```
    end process controller;
end BEHAVIOR;
-------------------------------------------------------------------------
-- Title      :   MAitrise
-------------------------------------------------------------------------
-- File       :   MSA_Z.vhd
-- Author     :   Aguibou BA
-- Company    :   PolySTIM
-- Last update:   2002/07/26

-------------------------------------------------------------------------
-- Description:   Unite de controle pour la mesure d'impedance
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/07/26  1.0      Guibs       Created
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.all;

entity msa_z is
    port (
                clk              : in std_logic;
                rst_n            : in std_logic;
                receving         : in std_logic;
                active           : in std_logic;
                eval             : in std_logic;
                ton              : in std_logic;
                toff             : in std_logic;
                test_z           : buffer std_logic;
                test_cal         : buffer std_logic;
                stim             :buffer std_logic

);
end msa_z;

architecture BEHAVIOR of msa_z is
      type STATES is (IDLE, CALIBRATION, ATTENTE1,
ATTENTE2,MESURE_Z,ATTENTE);

begin
      controller : process (clk, rst_n, receving,active,eval,ton,toff)

      variable state  : STATES;

begin  -- process controller
            -- activities triggered by asynchronous reset (active low)
            if rst_n = '0' then
                  state            := IDLE;
                  test_z           <= '0';
                  test_cal         <= '0';
```

```
                stim              <= '0';

            elsif clk'event and clk = '1' then
                case state is
-------------------------------------------------------------------------
-- IDLE
-------------------------------------------------------------------------
            when IDLE =>
                        if eval = '1' then
                              state             := CALIBRATION ;
                              test_z            <= '0';
                              test_cal          <= '1';
                              stim              <= '0';

                        elsif active = '1' then
                              state             := IDLE ;
                              test_z            <= '0';
                              test_cal          <= '0';
                              stim              <= '1';

                        else
                              state             := IDLE;
                              test_z            <= '0';
                              test_cal          <= '0';
                              stim              <= '0';

                        end if;
-------------------------------------------------------------------------
-- CALIBRATION
-------------------------------------------------------------------------
            when CALIBRATION =>
                  if receving = '1' then
                        if toff = '1' then
                              state             := ATTENTE1 ;
                              test_z            <= '0';
                              test_cal          <= '0';
                              stim              <= '0';

                        else
                              state             := CALIBRATION ;
                        end if;

                  else
                        state                   := IDLE;
                        test_z                  <= '0';
                        test_cal                <= '0';
                        stim                    <= '0';

                  end if;
-------------------------------------------------------------------------
-- ATTENTE1
-------------------------------------------------------------------------
            when ATTENTE1 =>
```

```
                        state                   := ATTENTE2 ;
                        test_z                  <= '0';
                        test_cal                <= '0';
                        stim                    <= '0';
-------------------------------------------------------------------------
-- ATTENTE2
-------------------------------------------------------------------------
            when ATTENTE2 =>
                        state                   := MESURE_Z ;
                        test_z                  <= '1';
                        test_cal                <= '0';
                        stim                    <= '0';
-------------------------------------------------------------------------
-- MESURE_Z
-------------------------------------------------------------------------
            when MESURE_Z =>
                  if receving = '1' then
                        if ton = '1' then
                              state             := ATTENTE;
                              test_z            <= '0';
                              test_cal          <= '0';
                              stim              <= '0';
                        else
                              state             := MESURE_Z ;
                        end if;

                  else
                        state                   := IDLE;
                        test_z                  <= '0';
                        test_cal                <= '0';
                        stim                    <= '0';

                  end if;

-------------------------------------------------------------------------
-- ATTENTE
-------------------------------------------------------------------------
            when ATTENTE =>
                  if receving = '1' then
                        if eval = '1' then
                              state             := ATTENTE;
                              test_z            <= '0';
                              test_cal          <= '0';
                              stim              <= '0';

                        else
                              state             := IDLE ;
                        end if;

                  else
                        state                   := IDLE;
                        test_z                  <= '0';
                        test_cal                <= '0';
```

```
                        stim                    <= '0';

                end if;

    end case;
end if;
end process controller;
end BEHAVIOR;
--------------------------------------------------------------------------
-- Ecole Polytechnique de Montreal
-- Groupe de Recherche en Microelectronique
-- PolyStim - Equipe de Recherche en Neurotechnologies
-- Version originale: Eric Schneider
--  Nom du module: Controleur
--  ce module génère les signaux de commande des grilles des transistors
--  des cicuits de permutation du courant dans les électrodes
-- Fichier: controleur.vhdl
--------------------------------------------------------------------------
--  Date Modification
--  98-11-24 Creation
--  02-08-10 Modification par Ba Aguibou
--------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity SIG_TRANS is
      port(
            test_cal          : in  std_logic;
            test_z            : in  std_logic;
            stim              : in  std_logic;
            up_down           : in  std_logic;
            active            : in  std_logic;
            ps1               : out std_logic;
            ps2               : out std_logic;
            psr1              : out std_logic;
            psr2              : out std_logic;
            ns1               : out std_logic;
            ns2               : out std_logic;
            nt1               : out std_logic;
            nt2               : out std_logic;
            t1                : out std_logic;
            pt1               : out std_logic;
            pt2               : out std_logic);
end SIG_TRANS ;

Architecture combinatoire of SIG_TRANS is

      signal local_ns1,local_ns2 : std_logic;
begin

      ps1 <= (not active) or ((not stim) or up_down) ;
      ps2 <= (not active) or ((not stim) or (not up_down));
```

```
      psr1 <= (not active) or ((not test_z) or up_down);
      psr2 <= (not active) or ((not test_z) or (not up_down));

      ns1 <= active and ((stim or test_z) and (not up_down)) ;
      ns2 <= active and ((stim or test_z) and ( up_down));

      nt1 <= active and (test_cal and up_down);
      nt2 <= active and (test_cal and (not up_down));

      pt1 <= (not active) or ((not test_cal) or (not up_down));
      pt2 <= (not active) or ((not test_cal) or (up_down));

      t1 <= (not active ) or (not test_cal);
end combinatoire;

-------------------------------------------------------------------------
-- File       : SWG_Z.vhd
-- Author     :   Guibs
-- Last update: 2002/04/17
-------------------------------------------------------------------------
-- Description: Ce module recoit les donnees en parallele de
-- DATA_RECOVERY et stimule adequatement les sorties up/down, cs et amp
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/04/17  1.0      venzz Created
-- Modifie par Ba aguibou le 10/08/2003
-------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity SWG_Z is

    port (clk : in std_logic;
        rst_n : in std_logic;
        enable : in std_logic;

-----------------------------------------------------------------------
-- INPUT: Parameters from data_recovery
-----------------------------------------------------------------------

        -- mode: indicates the signal chosen
        --  -> 0 for permanent
        --  -> 1 for selective
        mode : in std_logic;
        eval : in std_logic;

        -- Pulse period:
        --  -> hfp = high frequency period (ignored for permanent
stimulation)
        --  -> lfp = low frequency period
        lfp : in std_logic_vector(7 downto 0);
```

```
        hfp : in std_logic_vector(7 downto 0);

        -- Parameters for selective stimulation
        lfw : in std_logic_vector(7 downto 0);
        hfw : in std_logic_vector(7 downto 0);
        lfa : in std_logic_vector(7 downto 0);
        hfa : in std_logic_vector(7 downto 0);

        -- Parameters for permanent stimulation
        pton : in std_logic_vector(3 downto 0);
        ptoff : in std_logic_vector(3 downto 0);

-------------------------------------------------------------------------
-- OUTPUT: Sent to output devices
-------------------------------------------------------------------------

        ton_z : buffer std_logic;
        toff_z : buffer std_logic;

        -- Parameters for sign of the analog signal
        signe_swg  : buffer std_logic;

        -- Parameter for amplitude
        amp_swg : buffer std_logic_vector(7 downto 0);

        -- Parameter for chip select
        active_swg : buffer std_logic);

end SWG_Z;


architecture DATAFLOW of SWG_Z is
    component ctrl_unit
      port (clk : in std_logic;
            rst_n : in std_logic;
            hf_pulse : in std_logic;
            lf_pulse : in std_logic;
            busy : in std_logic;
            wsel : buffer std_logic;      -- 1 for high frequency, 0 for
low frequency
            start : buffer std_logic
);
    end component;

    component my_mux2a1
      generic (N : integer := 8);
       port (a, b : in std_logic_vector (N-1 downto 0);
        sel : in std_logic;
        q : buffer std_logic_vector (N-1 downto 0));
   end component;

    component wgen_n
      generic (N : integer := 8);
```

```
  port (clk : in std_logic;
        rst_n : in std_logic;
        enable : in std_logic;
        width : in std_logic_vector(N-1 downto 0);
        up : buffer std_logic;
        down : buffer std_logic);
end component;

component pgen_n_perm
  generic (N : integer := 8);
  port (clk : in std_logic;
        rst_n : in std_logic;
        enable : in std_logic;
        ack_n : in std_logic;
        pfreq : in std_logic_vector(N-1 downto 0);
        pulse : buffer std_logic);
end component;

component pgen_n_sel
  generic (N : integer := 8);
  port (clk : in std_logic;
        rst_n : in std_logic;
        enable : in std_logic;
        ack_n : in std_logic;
        pfreq : in std_logic_vector(N-1 downto 0);
        pulse : buffer std_logic);
end component;

component freq_divider
  generic (N : integer := 8);
  port (clk : in std_logic;
        rst_n : in std_logic;
        enable : in std_logic;
        clk_out : out std_logic);
end component;

component tontoff_z
  generic (N : integer := 4);
  port (clk : in std_logic;
        rst_n : in std_logic;
        enable : in std_logic;
        ton : in std_logic_vector(3 downto 0);
        toff : in std_logic_vector(3 downto 0);
          ton_z : buffer std_logic;
          toff_z : buffer std_logic;
        stim : buffer std_logic);
end component;

signal hf_pulse, lf_pulse : std_logic;
signal start : std_logic;
signal wsel : std_logic;
signal up,down : std_logic;
signal clk_262144 : std_logic;
```

```
    signal clk_4 : std_logic;
    signal clk_255 : std_logic;
    signal p_stim_on : std_logic;
    signal sel_en, perm_en : std_logic;
    signal stim : std_logic;
    signal mux_out : std_logic_vector(7 downto 0);
begin

-------------------------------------------------------------------------
-- Control unit
-------------------------------------------------------------------------
ctrl_unit_0 : ctrl_unit
      port map (clk, rst_n, hf_pulse, lf_pulse, active_swg, wsel,
start);

-------------------------------------------------------------------------
-- Mux 2 to 1 : Selects the width and Amplitude
-------------------------------------------------------------------------
       my_mux2a1_0 : my_mux2a1
            generic map (8)
            port map (lfw, hfw, wsel, mux_out);

    my_mux2a1_1 : my_mux2a1
            generic map (8)
            port map ( lfa, hfa, wsel, amp_swg);
-------------------------------------------------------------------------
-- Width generator


-------------------------------------------------------------------------
    wgen_n_0 : wgen_n
      generic map (8)
      port map (clk, start, stim, mux_out, up, down);

-------------------------------------------------------------------------
-- Chip select output
-------------------------------------------------------------------------
signe_swg <= up;
    active_swg <= up or down;

-------------------------------------------------------------------------
-- Pulse generator (High frequency: active for selective stimulation
only)
-------------------------------------------------------------------------
pgen_n_0 : pgen_n_sel
      generic map (8)
      port map (clk_4, rst_n, sel_en, start, hfp, hf_pulse);


-------------------------------------------------------------------------
-- Pulse generator (Low frequency)
-------------------------------------------------------------------------
pgen_n_1 : pgen_n_perm
```

```
      generic map (8)
      port map (clk_255, rst_n, stim, start, lfp, lf_pulse);


-------------------------------------------------------------------------
-- Frequency dividers
-------------------------------------------------------------------------
freq_divider_0 : freq_divider
      generic map (16)
      port map (clk, rst_n, perm_en, clk_262144);

    freq_divider_1 : freq_divider
      generic map (2)
      port map (clk, rst_n, sel_en, clk_4);

    freq_divider_2 : freq_divider
      generic map (8)
      port map (clk, rst_n, stim, clk_255);

-------------------------------------------------------------------------
-- TonToff
-------------------------------------------------------------------------
tontoff_0 : tontoff_z
      generic map (4)
      port map (clk_262144, rst_n, perm_en, pton, ptoff, ton_z,toff_z,
p_stim_on);

-------------------------------------------------------------------------
-- Stimulation control
-------------------------------------------------------------------------
    sel_en <= mode and enable and (not eval);
    perm_en <= (not mode or eval)  and enable;
    stim <= p_stim_on or sel_en or eval ;

end DATAFLOW;

-------------------------------------------------------------------------
-- Title      : M.Sc.A.
-- Project    : Implant Urinaire
-------------------------------------------------------------------------
-- File       : TOP.vhd
-- Author     : Aguibou BA
-- Last update: 2002/08/10
-------------------------------------------------------------------------
-- Description: Ce module decrit l'entite globale
-------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2002/08/10  1.0      GUIBS   Created
-------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
```

```
use IEEE.numeric_std.all;

entity TOP is
    port (
        clk             : in std_logic;
        rst_n           : in std_logic;

        ------------------------------------------------------------------
        -- Input Data
        ------------------------------------------------------------------
        data            : in std_logic;
        receiving       : in std_logic;
        my_manch_in     : in std_logic;
        my_rst_manch    : in std_logic;
        -- Data pour mesure impedance--
        f_osc_1         : in std_logic ;
        f_osc_2         : in std_logic ;
        test_clk        : in std_logic ;
        test_mode       : in std_logic ;
        sc_in           : in std_logic ;
        done            : out std_logic;
        serial          : out std_logic;
      -- DATA de Test functionnels
        valid           : buffer std_logic;
        crc_ok          : buffer std_logic;
        wr_n            : buffer std_logic;
        enab_gen1       : buffer std_logic;
        enab_gen2       : buffer std_logic;
        tk_ctrl_out1    : buffer std_logic;
        tk_ctrl_out2    : buffer std_logic;

------------------------------------------------------------------------
-- OUTPUT: Sent to output devices
------------------------------------------------------------------------
        my_data_out     : buffer std_logic;
        my_G_clock      : buffer std_logic;

        A_amp_1         : out std_logic_vector(7 downto 0);
        A_test_cal      : buffer std_logic;
        A_test_z        : buffer std_logic;
        A_ps1           : out std_logic;
        A_ps2           : out std_logic;
        A_psr1          : out std_logic;
        A_psr2          : out std_logic;
        A_ns1           : out std_logic;
        A_ns2           : out std_logic;
        A_nt1           : out std_logic;
        A_nt2           : out std_logic;
        A_t1            : out std_logic;
        A_pt1           : out std_logic;
        A_pt2           : out std_logic;

        B_amp_2         : out std_logic_vector(7 downto 0);
```

```
        B_test_cal      : buffer std_logic;
        B_test_z        : buffer std_logic;
        B_ps1           : out std_logic;
        B_ps2           : out std_logic;
        B_psr1          : out std_logic;
        B_psr2          : out std_logic;
        B_ns1           : out std_logic;
        B_ns2           : out std_logic;
        B_nt1           : out std_logic;
        B_nt2           : out std_logic;
        B_t1            : out std_logic;
        B_pt1           : out std_logic;
        B_pt2           : out std_logic
       );
end TOP;


architecture dataflow of TOP is
      component freq_estim
      port (
            clk         : in std_logic;
            rst_n       : in std_logic;
            test_z      : in std_logic;
            f_osc       : in std_logic;
            test_clk    : in std_logic;
            test_mode   : in std_logic;
            sc_in       : in std_logic;
            done        : out std_logic;
            serial      : out std_logic
            );
      end component;

      component my_inv
    port (
              D         : in STD_LOGIC;
              O         : buffer STD_LOGIC
            );
      end   component;

      component data_recovery
            port (
            clk         : in std_logic;
            rst_n       : in std_logic;
            data        : in std_logic;
            receiving   : in std_logic;
            mode_out    : buffer std_logic_vector(1 downto 0);
            channel     : buffer std_logic_vector(1 downto 0);
            addr        : buffer unsigned(4 downto 0);
            nb_pts      : buffer std_logic_vector(3 downto 0);
            data_out    : buffer std_logic_vector(9 downto 0);
            wr_n        : buffer std_logic;
            valid       : buffer std_logic;
            crc_ok      : buffer std_logic;
```

```
        tk_ctrl     : buffer std_logic);
   end component;

        component canal_z
  port (
        clk         : in std_logic;
        rst_n       : in std_logic;
        channel_in  : in std_logic;
        receiving   : in std_logic;
        mode_in     : in std_logic_vector(1 downto 0);
        addr_in     : in unsigned(4 downto 0);
        nb_pts_in   : in std_logic_vector(3 downto 0);
        data_in     : in std_logic_vector(7 downto 0);
        wr_n        : in std_logic;
        tk_ctrl_in  : in std_logic;
        tk_ctrl_out : buffer std_logic;
        enab_gen    : buffer std_logic;
        amp         : out std_logic_vector(7 downto 0);
        test_cal    : buffer std_logic;
        test_z      : buffer std_logic;
        ps1         : out std_logic;
        ps2         : out std_logic;
        psr1        : out std_logic;
        psr2        : out std_logic;
        ns1         : out std_logic;
        ns2         : out std_logic;
        nt1         : out std_logic;
        nt2         : out std_logic;
        t1          : out std_logic;
        pt1         : out std_logic;
        pt2         : out std_logic
        );
  end component ;

        component dec_manch
port (
        manch_in    : in std_logic;
        rst_manch_n : in std_logic;
        data_out    : buffer std_logic;
        G_clock     : buffer std_logic;
        rst_n       : in std_logic
        );
        end component;

        component mux2a1
     port (
        a, b        : in std_logic;
        sel         : in std_logic;
        q           : buffer std_logic
        );
        end component ;

-- Control signals
```

```
    signal chan_1,chan_2,tk_ctrl,channel, mesure_z,f_osc,sel_mux:
std_logic;
    signal chan_x   : std_logic_vector(1 downto 0);
    signal mode_out : std_logic_vector(1 downto 0);
    signal data_out : std_logic_vector(9 downto 0);
    signal nb_pts : std_logic_vector(3 downto 0);
    signal addr : unsigned(4 downto 0);
    signal data_in : std_logic_vector(7 downto 0);
    signal inv_clk1,inv_clk2 : std_logic;

begin

      my_inv1 : my_inv
            port map ( clk,inv_clk1);

      my_inv2 : my_inv
            port map ( inv_clk1,inv_clk2);

      dec_manch_0 : dec_manch
      port map (my_manch_in, my_rst_manch,my_data_out,my_G_clock,rst_n);

      data_recovery_0 :data_recovery
      port map
(inv_clk2,rst_n,data,receiving,mode_out,chan_x,addr,nb_pts,data_out,wr_n
,valid,crc_ok,tk_ctrl);

      data_in <=  data_out(9) & data_out(8) & data_out(7) & data_out(6)
& data_out(5) & data_out(4) & data_out(3) & data_out(2)  ;

      chan_1<= chan_x(0);
      chan_2<= chan_x(1);
      canal_1 : canal_Z
            port map (inv_clk2, rst_n, chan_1, receiving, mode_out,
addr, nb_pts, data_in, wr_n, tk_ctrl, tk_ctrl_out1, enab_gen1, A_amp_1,
A_test_cal, A_test_z, A_ps1, A_ps2, A_psr1, A_psr2, A_ns1, A_ns2, A_nt1,
A_nt2,A_t1,A_pt1,A_pt2);

      canal_2 : canal_Z
            port map
(inv_clk2,rst_n,chan_2,receiving,mode_out,addr,nb_pts,data_in,wr_n,tk_ct
rl,tk_ctrl_out2,enab_gen2,B_amp_2,B_test_cal,B_test_z,B_ps1,B_ps2,B_psr1
,B_psr2,B_ns1,B_ns2,B_nt1,B_nt2,B_t1,B_pt1,B_pt2);

      sel_mux <= A_test_cal or A_test_z;
            mux_f_osc : mux2a1
            port map (f_osc_2,f_osc_1, sel_mux,f_osc);
      mesure_z <= A_test_cal or A_test_z or B_test_cal or B_test_z;

      freq_estim_0 : freq_estim
            port map ( inv_clk2 , rst_n , mesure_z,f_osc, test_clk,
test_mode,sc_in,done,serial);

end dataflow;
```